

activemq-cpp-3.2.5

Generated by Doxygen 1.7.3

Tue Mar 8 2011 17:10:05



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>Data Structure Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Data Structure Index</b>	<b>25</b>
3.1	Data Structures . . . . .	25
<b>4</b>	<b>File Index</b>	<b>67</b>
4.1	File List . . . . .	67
<b>5</b>	<b>Namespace Documentation</b>	<b>93</b>
5.1	activemq Namespace Reference . . . . .	93
5.1.1	Detailed Description . . . . .	93
5.2	activemq::cmsutil Namespace Reference . . . . .	94
5.3	activemq::commands Namespace Reference . . . . .	95
5.4	activemq::core Namespace Reference . . . . .	96
5.5	activemq::core::policies Namespace Reference . . . . .	98
5.6	activemq::exceptions Namespace Reference . . . . .	98
5.7	activemq::io Namespace Reference . . . . .	98
5.8	activemq::library Namespace Reference . . . . .	98
5.9	activemq::state Namespace Reference . . . . .	98
5.10	activemq::threads Namespace Reference . . . . .	99
5.11	activemq::transport Namespace Reference . . . . .	99
5.12	activemq::transport::correlator Namespace Reference . . . . .	100
5.13	activemq::transport::failover Namespace Reference . . . . .	100
5.14	activemq::transport::inactivity Namespace Reference . . . . .	101
5.15	activemq::transport::logging Namespace Reference . . . . .	101
5.16	activemq::transport::mock Namespace Reference . . . . .	101
5.17	activemq::transport::tcp Namespace Reference . . . . .	102
5.18	activemq::util Namespace Reference . . . . .	102
5.19	activemq::wireformat Namespace Reference . . . . .	103
5.20	activemq::wireformat::openwire Namespace Reference . . . . .	103
5.21	activemq::wireformat::openwire::marshal Namespace Reference . . . . .	104
5.22	activemq::wireformat::openwire::marshal::v1 Namespace Reference . . . . .	104
5.23	activemq::wireformat::openwire::marshal::v2 Namespace Reference . . . . .	109
5.24	activemq::wireformat::openwire::marshal::v3 Namespace Reference . . . . .	113
5.25	activemq::wireformat::openwire::marshal::v4 Namespace Reference . . . . .	117

5.26	activemq::wireformat::openwire::marshal::v5 Namespace Reference . . . . .	121
5.27	activemq::wireformat::openwire::marshal::v6 Namespace Reference . . . . .	126
5.28	activemq::wireformat::openwire::utils Namespace Reference . . . . .	130
5.29	activemq::wireformat::stomp Namespace Reference . . . . .	130
5.30	cms Namespace Reference . . . . .	131
5.30.1	Detailed Description . . . . .	134
5.31	decaf Namespace Reference . . . . .	134
5.31.1	Detailed Description . . . . .	134
5.32	decaf::internal Namespace Reference . . . . .	135
5.33	decaf::internal::io Namespace Reference . . . . .	135
5.34	decaf::internal::net Namespace Reference . . . . .	135
5.35	decaf::internal::net::ssl Namespace Reference . . . . .	136
5.36	decaf::internal::net::ssl::openssl Namespace Reference . . . . .	136
5.37	decaf::internal::net::tcp Namespace Reference . . . . .	137
5.38	decaf::internal::nio Namespace Reference . . . . .	138
5.39	decaf::internal::security Namespace Reference . . . . .	138
5.40	decaf::internal::util Namespace Reference . . . . .	138
5.41	decaf::internal::util::concurrent Namespace Reference . . . . .	139
5.42	decaf::io Namespace Reference . . . . .	139
5.43	decaf::lang Namespace Reference . . . . .	141
5.43.1	Function Documentation . . . . .	143
5.43.1.1	operator!= . . . . .	143
5.43.1.2	operator!= . . . . .	143
5.43.1.3	operator!= . . . . .	143
5.43.1.4	operator!= . . . . .	144
5.43.1.5	operator== . . . . .	144
5.43.1.6	operator== . . . . .	144
5.43.1.7	operator== . . . . .	144
5.43.1.8	operator== . . . . .	144
5.44	decaf::lang::exceptions Namespace Reference . . . . .	144
5.45	decaf::net Namespace Reference . . . . .	145
5.46	decaf::net::ssl Namespace Reference . . . . .	146
5.47	decaf::nio Namespace Reference . . . . .	147
5.48	decaf::security Namespace Reference . . . . .	147
5.49	decaf::security::auth Namespace Reference . . . . .	148
5.50	decaf::security::auth::x500 Namespace Reference . . . . .	148
5.51	decaf::security::cert Namespace Reference . . . . .	148
5.52	decaf::util Namespace Reference . . . . .	149
5.53	decaf::util::comparators Namespace Reference . . . . .	151
5.54	decaf::util::concurrent Namespace Reference . . . . .	151
5.55	decaf::util::concurrent::atomic Namespace Reference . . . . .	153
5.56	decaf::util::concurrent::locks Namespace Reference . . . . .	153
5.57	decaf::util::logging Namespace Reference . . . . .	154
5.57.1	Enumeration Type Documentation . . . . .	155
5.57.1.1	Levels . . . . .	155
5.58	decaf::util::zip Namespace Reference . . . . .	156
5.59	std Namespace Reference . . . . .	157
<b>6</b>	<b>Data Structure Documentation</b>	<b>159</b>
6.1	decaf::util::AbstractCollection< E > Class Template Reference . . . . .	159



6.1.1	Detailed Description . . . . .	161
6.1.2	Constructor & Destructor Documentation . . . . .	162
6.1.2.1	AbstractCollection . . . . .	162
6.1.2.2	~AbstractCollection . . . . .	162
6.1.3	Member Function Documentation . . . . .	162
6.1.3.1	add . . . . .	162
6.1.3.2	addAll . . . . .	163
6.1.3.3	clear . . . . .	163
6.1.3.4	contains . . . . .	164
6.1.3.5	containsAll . . . . .	165
6.1.3.6	copy . . . . .	165
6.1.3.7	equals . . . . .	166
6.1.3.8	isEmpty . . . . .	166
6.1.3.9	lock . . . . .	167
6.1.3.10	notify . . . . .	167
6.1.3.11	notifyAll . . . . .	167
6.1.3.12	operator= . . . . .	168
6.1.3.13	remove . . . . .	168
6.1.3.14	removeAll . . . . .	169
6.1.3.15	retainAll . . . . .	170
6.1.3.16	toArray . . . . .	170
6.1.3.17	tryLock . . . . .	171
6.1.3.18	unlock . . . . .	171
6.1.3.19	wait . . . . .	171
6.1.3.20	wait . . . . .	172
6.1.3.21	wait . . . . .	172
6.1.4	Field Documentation . . . . .	173
6.1.4.1	mutex . . . . .	173
6.2	decaf::util::AbstractList< E > Class Template Reference . . . . .	173
6.2.1	Detailed Description . . . . .	173
6.2.2	Constructor & Destructor Documentation . . . . .	174
6.2.2.1	~AbstractList . . . . .	174
6.3	decaf::util::AbstractMap< K, V, COMPARATOR > Class Template Reference . . . . .	174
6.3.1	Detailed Description . . . . .	175
6.3.2	Constructor & Destructor Documentation . . . . .	175
6.3.2.1	~AbstractMap . . . . .	175
6.4	decaf::util::AbstractQueue< E > Class Template Reference . . . . .	175
6.4.1	Detailed Description . . . . .	176
6.4.2	Constructor & Destructor Documentation . . . . .	177
6.4.2.1	AbstractQueue . . . . .	177
6.4.2.2	~AbstractQueue . . . . .	177
6.4.3	Member Function Documentation . . . . .	177
6.4.3.1	add . . . . .	177
6.4.3.2	addAll . . . . .	177
6.4.3.3	clear . . . . .	178
6.4.3.4	element . . . . .	178
6.4.3.5	remove . . . . .	179
6.5	decaf::util::AbstractSequentialList< E > Class Template Reference . . . . .	179
6.5.1	Detailed Description . . . . .	180

6.5.2	Constructor & Destructor Documentation . . . . .	180
6.5.2.1	~AbstractSequentialList . . . . .	180
6.6	decaf::util::AbstractSet< E > Class Template Reference . . . . .	180
6.6.1	Detailed Description . . . . .	181
6.6.2	Constructor & Destructor Documentation . . . . .	181
6.6.2.1	~AbstractSet . . . . .	181
6.6.3	Member Function Documentation . . . . .	181
6.6.3.1	removeAll . . . . .	181
6.7	activemq::transport::AbstractTransportFactory Class Reference . . . . .	182
6.7.1	Detailed Description . . . . .	183
6.7.2	Constructor & Destructor Documentation . . . . .	183
6.7.2.1	~AbstractTransportFactory . . . . .	183
6.7.3	Member Function Documentation . . . . .	183
6.7.3.1	createWireFormat . . . . .	183
6.8	activemq::core::ActiveMQAckHandler Class Reference . . . . .	183
6.8.1	Detailed Description . . . . .	184
6.8.2	Constructor & Destructor Documentation . . . . .	184
6.8.2.1	~ActiveMQAckHandler . . . . .	184
6.8.3	Member Function Documentation . . . . .	184
6.8.3.1	acknowledgeMessage . . . . .	184
6.9	activemq::commands::ActiveMQBlobMessage Class Reference . . . . .	184
6.9.1	Constructor & Destructor Documentation . . . . .	186
6.9.1.1	ActiveMQBlobMessage . . . . .	186
6.9.1.2	~ActiveMQBlobMessage . . . . .	186
6.9.2	Member Function Documentation . . . . .	186
6.9.2.1	clone . . . . .	186
6.9.2.2	cloneDataStructure . . . . .	186
6.9.2.3	copyDataStructure . . . . .	186
6.9.2.4	equals . . . . .	187
6.9.2.5	getDataStructureType . . . . .	187
6.9.2.6	getMimeType . . . . .	187
6.9.2.7	getName . . . . .	187
6.9.2.8	getRemoteBlobUrl . . . . .	188
6.9.2.9	isDeletedByBroker . . . . .	188
6.9.2.10	setDeletedByBroker . . . . .	188
6.9.2.11	setMimeType . . . . .	188
6.9.2.12	setName . . . . .	188
6.9.2.13	setRemoteBlobUrl . . . . .	189
6.9.2.14	toString . . . . .	189
6.9.3	Field Documentation . . . . .	189
6.9.3.1	BINARY_MIME_TYPE . . . . .	189
6.9.3.2	ID_ACTIVEMQBLOBMESSAGE . . . . .	189
6.10	activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller Class Reference . . . . .	189
6.10.1	Detailed Description . . . . .	190
6.10.2	Constructor & Destructor Documentation . . . . .	191
6.10.2.1	ActiveMQBlobMessageMarshaller . . . . .	191
6.10.2.2	~ActiveMQBlobMessageMarshaller . . . . .	191
6.10.3	Member Function Documentation . . . . .	191
6.10.3.1	createObject . . . . .	191

6.10.3.2	getDataStructureType . . . . .	191
6.10.3.3	looseMarshal . . . . .	191
6.10.3.4	looseUnmarshal . . . . .	192
6.10.3.5	tightMarshal1 . . . . .	192
6.10.3.6	tightMarshal2 . . . . .	193
6.10.3.7	tightUnmarshal . . . . .	193
6.11	activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller	
	Class Reference . . . . .	194
6.11.1	Detailed Description . . . . .	195
6.11.2	Constructor & Destructor Documentation . . . . .	195
6.11.2.1	ActiveMQBlobMessageMarshaller . . . . .	195
6.11.2.2	~ActiveMQBlobMessageMarshaller . . . . .	195
6.11.3	Member Function Documentation . . . . .	195
6.11.3.1	createObject . . . . .	195
6.11.3.2	getDataStructureType . . . . .	195
6.11.3.3	looseMarshal . . . . .	195
6.11.3.4	looseUnmarshal . . . . .	196
6.11.3.5	tightMarshal1 . . . . .	196
6.11.3.6	tightMarshal2 . . . . .	197
6.11.3.7	tightUnmarshal . . . . .	197
6.12	activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller	
	Class Reference . . . . .	198
6.12.1	Detailed Description . . . . .	199
6.12.2	Constructor & Destructor Documentation . . . . .	199
6.12.2.1	ActiveMQBlobMessageMarshaller . . . . .	199
6.12.2.2	~ActiveMQBlobMessageMarshaller . . . . .	199
6.12.3	Member Function Documentation . . . . .	199
6.12.3.1	createObject . . . . .	199
6.12.3.2	getDataStructureType . . . . .	199
6.12.3.3	looseMarshal . . . . .	200
6.12.3.4	looseUnmarshal . . . . .	200
6.12.3.5	tightMarshal1 . . . . .	201
6.12.3.6	tightMarshal2 . . . . .	201
6.12.3.7	tightUnmarshal . . . . .	202
6.13	activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller	
	Class Reference . . . . .	202
6.13.1	Detailed Description . . . . .	203
6.13.2	Constructor & Destructor Documentation . . . . .	203
6.13.2.1	ActiveMQBlobMessageMarshaller . . . . .	203
6.13.2.2	~ActiveMQBlobMessageMarshaller . . . . .	203
6.13.3	Member Function Documentation . . . . .	203
6.13.3.1	createObject . . . . .	203
6.13.3.2	getDataStructureType . . . . .	204
6.13.3.3	looseMarshal . . . . .	204
6.13.3.4	looseUnmarshal . . . . .	204
6.13.3.5	tightMarshal1 . . . . .	205
6.13.3.6	tightMarshal2 . . . . .	205
6.13.3.7	tightUnmarshal . . . . .	206
6.14	activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller	
	Class Reference . . . . .	206

6.14.1	Detailed Description . . . . .	207
6.14.2	Constructor & Destructor Documentation . . . . .	208
6.14.2.1	ActiveMQBlobMessageMarshaller . . . . .	208
6.14.2.2	~ActiveMQBlobMessageMarshaller . . . . .	208
6.14.3	Member Function Documentation . . . . .	208
6.14.3.1	createObject . . . . .	208
6.14.3.2	getDataStructureType . . . . .	208
6.14.3.3	looseMarshal . . . . .	208
6.14.3.4	looseUnmarshal . . . . .	209
6.14.3.5	tightMarshal1 . . . . .	209
6.14.3.6	tightMarshal2 . . . . .	210
6.14.3.7	tightUnmarshal . . . . .	210
6.15	activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller Class Reference . . . . .	211
6.15.1	Detailed Description . . . . .	212
6.15.2	Constructor & Destructor Documentation . . . . .	212
6.15.2.1	ActiveMQBlobMessageMarshaller . . . . .	212
6.15.2.2	~ActiveMQBlobMessageMarshaller . . . . .	212
6.15.3	Member Function Documentation . . . . .	212
6.15.3.1	createObject . . . . .	212
6.15.3.2	getDataStructureType . . . . .	212
6.15.3.3	looseMarshal . . . . .	212
6.15.3.4	looseUnmarshal . . . . .	213
6.15.3.5	tightMarshal1 . . . . .	213
6.15.3.6	tightMarshal2 . . . . .	214
6.15.3.7	tightUnmarshal . . . . .	214
6.16	activemq::commands::ActiveMQBytesMessage Class Reference . . . . .	215
6.16.1	Constructor & Destructor Documentation . . . . .	219
6.16.1.1	ActiveMQBytesMessage . . . . .	219
6.16.1.2	~ActiveMQBytesMessage . . . . .	219
6.16.2	Member Function Documentation . . . . .	219
6.16.2.1	clearBody . . . . .	219
6.16.2.2	clone . . . . .	219
6.16.2.3	cloneDataStructure . . . . .	219
6.16.2.4	copyDataStructure . . . . .	219
6.16.2.5	equals . . . . .	220
6.16.2.6	getBodyBytes . . . . .	220
6.16.2.7	getBodyLength . . . . .	221
6.16.2.8	getDataStructureType . . . . .	221
6.16.2.9	onSend . . . . .	221
6.16.2.10	readBoolean . . . . .	221
6.16.2.11	readByte . . . . .	222
6.16.2.12	readBytes . . . . .	222
6.16.2.13	readBytes . . . . .	223
6.16.2.14	readChar . . . . .	224
6.16.2.15	readDouble . . . . .	224
6.16.2.16	readFloat . . . . .	225
6.16.2.17	readInt . . . . .	225
6.16.2.18	readLong . . . . .	226
6.16.2.19	readShort . . . . .	226

6.16.2.20	readString	227
6.16.2.21	readUnsignedShort	227
6.16.2.22	readUTF	228
6.16.2.23	reset	228
6.16.2.24	setBodyBytes	228
6.16.2.25	toString	229
6.16.2.26	writeBoolean	229
6.16.2.27	writeByte	229
6.16.2.28	writeBytes	230
6.16.2.29	writeBytes	230
6.16.2.30	writeChar	231
6.16.2.31	writeDouble	231
6.16.2.32	writeFloat	232
6.16.2.33	writeInt	232
6.16.2.34	writeLong	232
6.16.2.35	writeShort	233
6.16.2.36	writeString	233
6.16.2.37	writeUnsignedShort	234
6.16.2.38	writeUTF	234
6.16.3	Field Documentation	234
6.16.3.1	ID_ACTIVEMQBYTESMESSAGE	234
6.17	activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller	
	Class Reference	235
6.17.1	Detailed Description	236
6.17.2	Constructor & Destructor Documentation	236
6.17.2.1	ActiveMQBytesMessageMarshaller	236
6.17.2.2	~ActiveMQBytesMessageMarshaller	236
6.17.3	Member Function Documentation	236
6.17.3.1	createObject	236
6.17.3.2	getDataStructureType	236
6.17.3.3	looseMarshal	236
6.17.3.4	looseUnmarshal	237
6.17.3.5	tightMarshal1	237
6.17.3.6	tightMarshal2	238
6.17.3.7	tightUnmarshal	238
6.18	activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller	
	Class Reference	239
6.18.1	Detailed Description	240
6.18.2	Constructor & Destructor Documentation	240
6.18.2.1	ActiveMQBytesMessageMarshaller	240
6.18.2.2	~ActiveMQBytesMessageMarshaller	240
6.18.3	Member Function Documentation	240
6.18.3.1	createObject	240
6.18.3.2	getDataStructureType	240
6.18.3.3	looseMarshal	241
6.18.3.4	looseUnmarshal	241
6.18.3.5	tightMarshal1	242
6.18.3.6	tightMarshal2	242
6.18.3.7	tightUnmarshal	243

6.19	activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller	
	Class Reference . . . . .	243
6.19.1	Detailed Description . . . . .	244
6.19.2	Constructor & Destructor Documentation . . . . .	244
	6.19.2.1 ActiveMQBytesMessageMarshaller . . . . .	244
	6.19.2.2 ~ActiveMQBytesMessageMarshaller . . . . .	244
6.19.3	Member Function Documentation . . . . .	244
	6.19.3.1 createObject . . . . .	244
	6.19.3.2 getDataStructureType . . . . .	245
	6.19.3.3 looseMarshal . . . . .	245
	6.19.3.4 looseUnmarshal . . . . .	245
	6.19.3.5 tightMarshal1 . . . . .	246
	6.19.3.6 tightMarshal2 . . . . .	246
	6.19.3.7 tightUnmarshal . . . . .	247
6.20	activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller	
	Class Reference . . . . .	247
6.20.1	Detailed Description . . . . .	248
6.20.2	Constructor & Destructor Documentation . . . . .	249
	6.20.2.1 ActiveMQBytesMessageMarshaller . . . . .	249
	6.20.2.2 ~ActiveMQBytesMessageMarshaller . . . . .	249
6.20.3	Member Function Documentation . . . . .	249
	6.20.3.1 createObject . . . . .	249
	6.20.3.2 getDataStructureType . . . . .	249
	6.20.3.3 looseMarshal . . . . .	249
	6.20.3.4 looseUnmarshal . . . . .	250
	6.20.3.5 tightMarshal1 . . . . .	250
	6.20.3.6 tightMarshal2 . . . . .	251
	6.20.3.7 tightUnmarshal . . . . .	251
6.21	activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller	
	Class Reference . . . . .	252
6.21.1	Detailed Description . . . . .	253
6.21.2	Constructor & Destructor Documentation . . . . .	253
	6.21.2.1 ActiveMQBytesMessageMarshaller . . . . .	253
	6.21.2.2 ~ActiveMQBytesMessageMarshaller . . . . .	253
6.21.3	Member Function Documentation . . . . .	253
	6.21.3.1 createObject . . . . .	253
	6.21.3.2 getDataStructureType . . . . .	253
	6.21.3.3 looseMarshal . . . . .	253
	6.21.3.4 looseUnmarshal . . . . .	254
	6.21.3.5 tightMarshal1 . . . . .	254
	6.21.3.6 tightMarshal2 . . . . .	255
	6.21.3.7 tightUnmarshal . . . . .	255
6.22	activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller	
	Class Reference . . . . .	256
6.22.1	Detailed Description . . . . .	257
6.22.2	Constructor & Destructor Documentation . . . . .	257
	6.22.2.1 ActiveMQBytesMessageMarshaller . . . . .	257
	6.22.2.2 ~ActiveMQBytesMessageMarshaller . . . . .	257
6.22.3	Member Function Documentation . . . . .	257
	6.22.3.1 createObject . . . . .	257

6.22.3.2	getDataStructureType . . . . .	257
6.22.3.3	looseMarshal . . . . .	258
6.22.3.4	looseUnmarshal . . . . .	258
6.22.3.5	tightMarshal1 . . . . .	259
6.22.3.6	tightMarshal2 . . . . .	259
6.22.3.7	tightUnmarshal . . . . .	260
6.23	activemq::core::ActiveMQConnection Class Reference . . . . .	260
6.23.1	Detailed Description . . . . .	266
6.23.2	Constructor & Destructor Documentation . . . . .	266
6.23.2.1	ActiveMQConnection . . . . .	266
6.23.2.2	~ActiveMQConnection . . . . .	266
6.23.3	Member Function Documentation . . . . .	266
6.23.3.1	addDispatcher . . . . .	266
6.23.3.2	addProducer . . . . .	267
6.23.3.3	addTransportListener . . . . .	267
6.23.3.4	close . . . . .	267
6.23.3.5	createSession . . . . .	267
6.23.3.6	createSession . . . . .	268
6.23.3.7	destroyDestination . . . . .	268
6.23.3.8	destroyDestination . . . . .	269
6.23.3.9	fire . . . . .	269
6.23.3.10	getBrokerURL . . . . .	269
6.23.3.11	getClientID . . . . .	270
6.23.3.12	getCloseTimeout . . . . .	270
6.23.3.13	getConnectionId . . . . .	270
6.23.3.14	getConnectionInfo . . . . .	270
6.23.3.15	getExceptionListener . . . . .	270
6.23.3.16	getMetaData . . . . .	271
6.23.3.17	getNextLocalTransactionId . . . . .	271
6.23.3.18	getNextSessionId . . . . .	271
6.23.3.19	getNextTempDestinationId . . . . .	271
6.23.3.20	getPassword . . . . .	272
6.23.3.21	getPrefetchPolicy . . . . .	272
6.23.3.22	getProducerWindowSize . . . . .	272
6.23.3.23	getRedeliveryPolicy . . . . .	272
6.23.3.24	getSendTimeout . . . . .	272
6.23.3.25	getTransport . . . . .	273
6.23.3.26	getUsername . . . . .	273
6.23.3.27	isAlwaysSyncSend . . . . .	273
6.23.3.28	isClosed . . . . .	273
6.23.3.29	isDispatchAsync . . . . .	273
6.23.3.30	isStarted . . . . .	273
6.23.3.31	isTransportFailed . . . . .	274
6.23.3.32	isUseAsyncSend . . . . .	274
6.23.3.33	isUseCompression . . . . .	274
6.23.3.34	onCommand . . . . .	274
6.23.3.35	oneway . . . . .	274
6.23.3.36	onException . . . . .	275
6.23.3.37	removeDispatcher . . . . .	275
6.23.3.38	removeProducer . . . . .	275

6.23.3.39	removeSession . . . . .	275
6.23.3.40	removeTransportListener . . . . .	276
6.23.3.41	sendPullRequest . . . . .	276
6.23.3.42	setAlwaysSyncSend . . . . .	276
6.23.3.43	setBrokerURL . . . . .	276
6.23.3.44	setClientID . . . . .	277
6.23.3.45	setCloseTimeout . . . . .	277
6.23.3.46	setDefaultClientId . . . . .	277
6.23.3.47	setDispatchAsync . . . . .	278
6.23.3.48	setExceptionListener . . . . .	278
6.23.3.49	setPassword . . . . .	278
6.23.3.50	setPrefetchPolicy . . . . .	278
6.23.3.51	setProducerWindowSize . . . . .	279
6.23.3.52	setRedeliveryPolicy . . . . .	279
6.23.3.53	setSendTimeout . . . . .	279
6.23.3.54	setTransportInterruptionProcessingComplete . . . . .	279
6.23.3.55	setUseAsyncSend . . . . .	279
6.23.3.56	setUseCompression . . . . .	280
6.23.3.57	setUsername . . . . .	280
6.23.3.58	start . . . . .	280
6.23.3.59	stop . . . . .	280
6.23.3.60	syncRequest . . . . .	280
6.23.3.61	transportInterrupted . . . . .	281
6.23.3.62	transportResumed . . . . .	281
6.24	activemq::core::ActiveMQConnectionFactory Class Reference . . . . .	281
6.24.1	Constructor & Destructor Documentation . . . . .	284
6.24.1.1	ActiveMQConnectionFactory . . . . .	284
6.24.1.2	ActiveMQConnectionFactory . . . . .	284
6.24.1.3	~ActiveMQConnectionFactory . . . . .	285
6.24.2	Member Function Documentation . . . . .	285
6.24.2.1	createConnection . . . . .	285
6.24.2.2	createConnection . . . . .	285
6.24.2.3	createConnection . . . . .	286
6.24.2.4	createConnection . . . . .	286
6.24.2.5	getBrokerURL . . . . .	287
6.24.2.6	getClientId . . . . .	287
6.24.2.7	getCloseTimeout . . . . .	287
6.24.2.8	getExceptionListener . . . . .	287
6.24.2.9	getPassword . . . . .	287
6.24.2.10	getPrefetchPolicy . . . . .	288
6.24.2.11	getProducerWindowSize . . . . .	288
6.24.2.12	getRedeliveryPolicy . . . . .	288
6.24.2.13	getSendTimeout . . . . .	288
6.24.2.14	getUsername . . . . .	288
6.24.2.15	isAlwaysSyncSend . . . . .	289
6.24.2.16	isDispatchAsync . . . . .	289
6.24.2.17	isUseAsyncSend . . . . .	289
6.24.2.18	isUseCompression . . . . .	289
6.24.2.19	setAlwaysSyncSend . . . . .	289
6.24.2.20	setBrokerURL . . . . .	290



6.24.2.21	setClientId	290
6.24.2.22	setCloseTimeout	290
6.24.2.23	setDispatchAsync	290
6.24.2.24	setExceptionListener	290
6.24.2.25	setPassword	291
6.24.2.26	setPrefetchPolicy	291
6.24.2.27	setProducerWindowSize	291
6.24.2.28	setRedeliveryPolicy	291
6.24.2.29	setSendTimeout	292
6.24.2.30	setUseAsyncSend	292
6.24.2.31	setUseCompression	292
6.24.2.32	setUsername	292
6.24.3	Field Documentation	293
6.24.3.1	DEFAULT_URI	293
6.25	activemq::core::ActiveMQConnectionMetaData Class Reference	293
6.25.1	Detailed Description	294
6.25.2	Constructor & Destructor Documentation	294
6.25.2.1	ActiveMQConnectionMetaData	294
6.25.2.2	~ActiveMQConnectionMetaData	294
6.25.3	Member Function Documentation	294
6.25.3.1	getCMSMajorVersion	294
6.25.3.2	getCMSMinorVersion	294
6.25.3.3	getCMSProviderName	295
6.25.3.4	getCMSVersion	295
6.25.3.5	getCMSXPropertyNames	295
6.25.3.6	getProviderMajorVersion	296
6.25.3.7	getProviderMinorVersion	296
6.25.3.8	getProviderVersion	296
6.26	activemq::core::ActiveMQConstants Class Reference	297
6.26.1	Detailed Description	298
6.26.2	Member Enumeration Documentation	298
6.26.2.1	AckType	298
6.26.2.2	DestinationActions	298
6.26.2.3	DestinationOption	299
6.26.2.4	TransactionState	299
6.26.2.5	URIParam	299
6.26.3	Member Function Documentation	300
6.26.3.1	toDestinationOption	300
6.26.3.2	toString	300
6.26.3.3	toString	300
6.26.3.4	toURIOption	300
6.27	activemq::core::ActiveMQConsumer Class Reference	300
6.27.1	Constructor & Destructor Documentation	303
6.27.1.1	ActiveMQConsumer	303
6.27.1.2	~ActiveMQConsumer	303
6.27.2	Member Function Documentation	303
6.27.2.1	acknowledge	303
6.27.2.2	acknowledge	304
6.27.2.3	afterMessageIsConsumed	304
6.27.2.4	beforeMessageIsConsumed	304

6.27.2.5	clearMessagesInProgress	304
6.27.2.6	close	304
6.27.2.7	commit	305
6.27.2.8	deliverAcks	305
6.27.2.9	dequeue	305
6.27.2.10	dispatch	305
6.27.2.11	doClose	306
6.27.2.12	getConsumerId	306
6.27.2.13	getConsumerInfo	306
6.27.2.14	getLastDeliveredSequenceId	306
6.27.2.15	getMessageAvailableCount	306
6.27.2.16	getMessageListener	307
6.27.2.17	getMessageSelector	307
6.27.2.18	getRedeliveryPolicy	307
6.27.2.19	inProgressClearRequired	307
6.27.2.20	isClosed	307
6.27.2.21	isSynchronizationRegistered	308
6.27.2.22	iterate	308
6.27.2.23	receive	308
6.27.2.24	receive	308
6.27.2.25	receiveNoWait	309
6.27.2.26	rollback	309
6.27.2.27	setLastDeliveredSequenceId	309
6.27.2.28	setMessageListener	309
6.27.2.29	setRedeliveryPolicy	310
6.27.2.30	setSynchronizationRegistered	310
6.27.2.31	start	310
6.27.2.32	stop	310
6.28	activemq::library::ActiveMQCPP Class Reference	310
6.28.1	Constructor & Destructor Documentation	311
6.28.1.1	ActiveMQCPP	311
6.28.1.2	ActiveMQCPP	311
6.28.1.3	~ActiveMQCPP	311
6.28.2	Member Function Documentation	311
6.28.2.1	initializeLibrary	311
6.28.2.2	initializeLibrary	312
6.28.2.3	operator=	312
6.28.2.4	shutdownLibrary	312
6.29	activemq::commands::ActiveMQDestination Class Reference	312
6.29.1	Constructor & Destructor Documentation	315
6.29.1.1	ActiveMQDestination	315
6.29.1.2	ActiveMQDestination	315
6.29.1.3	~ActiveMQDestination	315
6.29.2	Member Function Documentation	315
6.29.2.1	cloneDataStructure	315
6.29.2.2	copyDataStructure	316
6.29.2.3	createDestination	316
6.29.2.4	createTemporaryName	316
6.29.2.5	equals	317
6.29.2.6	getClientId	317

6.29.2.7	getCMSDestination . . . . .	317
6.29.2.8	getDataStructureType . . . . .	317
6.29.2.9	getDestinationType . . . . .	318
6.29.2.10	getOptions . . . . .	318
6.29.2.11	getOrderedTarget . . . . .	318
6.29.2.12	getPhysicalName . . . . .	318
6.29.2.13	getPhysicalName . . . . .	319
6.29.2.14	isAdvisory . . . . .	319
6.29.2.15	isComposite . . . . .	319
6.29.2.16	isConnectionAdvisory . . . . .	319
6.29.2.17	isConsumerAdvisory . . . . .	319
6.29.2.18	isExclusive . . . . .	319
6.29.2.19	isOrdered . . . . .	319
6.29.2.20	isProducerAdvisory . . . . .	320
6.29.2.21	isQueue . . . . .	320
6.29.2.22	isTemporary . . . . .	320
6.29.2.23	isTopic . . . . .	320
6.29.2.24	isWildcard . . . . .	320
6.29.2.25	setAdvisory . . . . .	321
6.29.2.26	setExclusive . . . . .	321
6.29.2.27	setOrdered . . . . .	321
6.29.2.28	setOrderedTarget . . . . .	321
6.29.2.29	setPhysicalName . . . . .	321
6.29.2.30	toString . . . . .	321
6.29.3	Field Documentation . . . . .	322
6.29.3.1	advisory . . . . .	322
6.29.3.2	ADVISORY_PREFIX . . . . .	322
6.29.3.3	COMPOSITE_SEPARATOR . . . . .	322
6.29.3.4	CONNECTION_ADVISORY_PREFIX . . . . .	322
6.29.3.5	CONSUMER_ADVISORY_PREFIX . . . . .	322
6.29.3.6	DEFAULT_ORDERED_TARGET . . . . .	322
6.29.3.7	exclusive . . . . .	323
6.29.3.8	ID_ACTIVEMQDESTINATION . . . . .	323
6.29.3.9	options . . . . .	323
6.29.3.10	ordered . . . . .	323
6.29.3.11	orderedTarget . . . . .	323
6.29.3.12	physicalName . . . . .	323
6.29.3.13	PRODUCER_ADVISORY_PREFIX . . . . .	323
6.29.3.14	QUEUE_QUALIFIED_PREFIX . . . . .	323
6.29.3.15	TEMP_POSTFIX . . . . .	323
6.29.3.16	TEMP_PREFIX . . . . .	323
6.29.3.17	TEMP_QUEUE_QUALIFIED_PREFIX . . . . .	323
6.29.3.18	TEMP_TOPIC_QUALIFIED_PREFIX . . . . .	323
6.29.3.19	TOPIC_QUALIFIED_PREFIX . . . . .	323
6.30	activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller	
	Class Reference . . . . .	324
6.30.1	Detailed Description . . . . .	324
6.30.2	Constructor & Destructor Documentation . . . . .	325
6.30.2.1	ActiveMQDestinationMarshaller . . . . .	325
6.30.2.2	~ActiveMQDestinationMarshaller . . . . .	325

6.30.3	Member Function Documentation . . . . .	325
6.30.3.1	looseMarshal . . . . .	325
6.30.3.2	looseUnmarshal . . . . .	325
6.30.3.3	tightMarshal1 . . . . .	326
6.30.3.4	tightMarshal2 . . . . .	326
6.30.3.5	tightUnmarshal . . . . .	327
6.31	activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller	
	Class Reference . . . . .	328
6.31.1	Detailed Description . . . . .	328
6.31.2	Constructor & Destructor Documentation . . . . .	329
6.31.2.1	ActiveMQDestinationMarshaller . . . . .	329
6.31.2.2	~ActiveMQDestinationMarshaller . . . . .	329
6.31.3	Member Function Documentation . . . . .	329
6.31.3.1	looseMarshal . . . . .	329
6.31.3.2	looseUnmarshal . . . . .	329
6.31.3.3	tightMarshal1 . . . . .	330
6.31.3.4	tightMarshal2 . . . . .	330
6.31.3.5	tightUnmarshal . . . . .	331
6.32	activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	
	Class Reference . . . . .	332
6.32.1	Detailed Description . . . . .	332
6.32.2	Constructor & Destructor Documentation . . . . .	333
6.32.2.1	ActiveMQDestinationMarshaller . . . . .	333
6.32.2.2	~ActiveMQDestinationMarshaller . . . . .	333
6.32.3	Member Function Documentation . . . . .	333
6.32.3.1	looseMarshal . . . . .	333
6.32.3.2	looseUnmarshal . . . . .	333
6.32.3.3	tightMarshal1 . . . . .	334
6.32.3.4	tightMarshal2 . . . . .	334
6.32.3.5	tightUnmarshal . . . . .	335
6.33	activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller	
	Class Reference . . . . .	336
6.33.1	Detailed Description . . . . .	336
6.33.2	Constructor & Destructor Documentation . . . . .	337
6.33.2.1	ActiveMQDestinationMarshaller . . . . .	337
6.33.2.2	~ActiveMQDestinationMarshaller . . . . .	337
6.33.3	Member Function Documentation . . . . .	337
6.33.3.1	looseMarshal . . . . .	337
6.33.3.2	looseUnmarshal . . . . .	337
6.33.3.3	tightMarshal1 . . . . .	338
6.33.3.4	tightMarshal2 . . . . .	338
6.33.3.5	tightUnmarshal . . . . .	339
6.34	activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller	
	Class Reference . . . . .	340
6.34.1	Detailed Description . . . . .	340
6.34.2	Constructor & Destructor Documentation . . . . .	341
6.34.2.1	ActiveMQDestinationMarshaller . . . . .	341
6.34.2.2	~ActiveMQDestinationMarshaller . . . . .	341
6.34.3	Member Function Documentation . . . . .	341
6.34.3.1	looseMarshal . . . . .	341

6.34.3.2	looseUnmarshal . . . . .	341
6.34.3.3	tightMarshal1 . . . . .	342
6.34.3.4	tightMarshal2 . . . . .	342
6.34.3.5	tightUnmarshal . . . . .	343
6.35	activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller	
	Class Reference . . . . .	344
6.35.1	Detailed Description . . . . .	344
6.35.2	Constructor & Destructor Documentation . . . . .	345
6.35.2.1	ActiveMQDestinationMarshaller . . . . .	345
6.35.2.2	~ActiveMQDestinationMarshaller . . . . .	345
6.35.3	Member Function Documentation . . . . .	345
6.35.3.1	looseMarshal . . . . .	345
6.35.3.2	looseUnmarshal . . . . .	345
6.35.3.3	tightMarshal1 . . . . .	346
6.35.3.4	tightMarshal2 . . . . .	346
6.35.3.5	tightUnmarshal . . . . .	347
6.36	activemq::exceptions::ActiveMQException Class Reference . . . . .	348
6.36.1	Constructor & Destructor Documentation . . . . .	348
6.36.1.1	ActiveMQException . . . . .	348
6.36.1.2	ActiveMQException . . . . .	348
6.36.1.3	ActiveMQException . . . . .	349
6.36.1.4	ActiveMQException . . . . .	349
6.36.1.5	~ActiveMQException . . . . .	349
6.36.2	Member Function Documentation . . . . .	349
6.36.2.1	clone . . . . .	349
6.36.2.2	convertToCMSException . . . . .	349
6.37	activemq::commands::ActiveMQMapMessage Class Reference . . . . .	350
6.37.1	Constructor & Destructor Documentation . . . . .	353
6.37.1.1	ActiveMQMapMessage . . . . .	353
6.37.1.2	~ActiveMQMapMessage . . . . .	353
6.37.2	Member Function Documentation . . . . .	353
6.37.2.1	beforeMarshal . . . . .	353
6.37.2.2	checkMapIsUnmarshalled . . . . .	353
6.37.2.3	clearBody . . . . .	354
6.37.2.4	clone . . . . .	354
6.37.2.5	cloneDataStructure . . . . .	354
6.37.2.6	copyDataStructure . . . . .	354
6.37.2.7	equals . . . . .	355
6.37.2.8	getBoolean . . . . .	355
6.37.2.9	getByte . . . . .	355
6.37.2.10	getBytes . . . . .	356
6.37.2.11	getChar . . . . .	356
6.37.2.12	getDataStructureType . . . . .	356
6.37.2.13	getDouble . . . . .	357
6.37.2.14	getFloat . . . . .	357
6.37.2.15	getInt . . . . .	357
6.37.2.16	getLong . . . . .	358
6.37.2.17	getMap . . . . .	358
6.37.2.18	getMap . . . . .	358
6.37.2.19	getMapNames . . . . .	358

6.37.2.20	getShort	359
6.37.2.21	getString	359
6.37.2.22	isMarshalAware	359
6.37.2.23	itemExists	360
6.37.2.24	setBoolean	360
6.37.2.25	setByte	360
6.37.2.26	setBytes	361
6.37.2.27	setChar	361
6.37.2.28	setDouble	362
6.37.2.29	setFloat	362
6.37.2.30	setInt	362
6.37.2.31	setLong	363
6.37.2.32	setShort	363
6.37.2.33	setString	364
6.37.2.34	toString	364
6.37.3	Field Documentation	364
6.37.3.1	ID_ACTIVEMQMAPMESSAGE	364
6.38	activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller	
	Class Reference	364
6.38.1	Detailed Description	365
6.38.2	Constructor & Destructor Documentation	366
6.38.2.1	ActiveMQMapMessageMarshaller	366
6.38.2.2	~ActiveMQMapMessageMarshaller	366
6.38.3	Member Function Documentation	366
6.38.3.1	createObject	366
6.38.3.2	getDataStructureType	366
6.38.3.3	looseMarshal	366
6.38.3.4	looseUnmarshal	367
6.38.3.5	tightMarshal1	367
6.38.3.6	tightMarshal2	368
6.38.3.7	tightUnmarshal	368
6.39	activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller	
	Class Reference	369
6.39.1	Detailed Description	370
6.39.2	Constructor & Destructor Documentation	370
6.39.2.1	ActiveMQMapMessageMarshaller	370
6.39.2.2	~ActiveMQMapMessageMarshaller	370
6.39.3	Member Function Documentation	370
6.39.3.1	createObject	370
6.39.3.2	getDataStructureType	370
6.39.3.3	looseMarshal	370
6.39.3.4	looseUnmarshal	371
6.39.3.5	tightMarshal1	371
6.39.3.6	tightMarshal2	372
6.39.3.7	tightUnmarshal	372
6.40	activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller	
	Class Reference	373
6.40.1	Detailed Description	374
6.40.2	Constructor & Destructor Documentation	374
6.40.2.1	ActiveMQMapMessageMarshaller	374

6.40.2.2	~ActiveMQMapMessageMarshaller . . . . .	374
6.40.3	Member Function Documentation . . . . .	374
6.40.3.1	createObject . . . . .	374
6.40.3.2	getDataStructureType . . . . .	374
6.40.3.3	looseMarshal . . . . .	375
6.40.3.4	looseUnmarshal . . . . .	375
6.40.3.5	tightMarshal1 . . . . .	376
6.40.3.6	tightMarshal2 . . . . .	376
6.40.3.7	tightUnmarshal . . . . .	377
6.41	activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	
	Class Reference . . . . .	377
6.41.1	Detailed Description . . . . .	378
6.41.2	Constructor & Destructor Documentation . . . . .	378
6.41.2.1	ActiveMQMapMessageMarshaller . . . . .	378
6.41.2.2	~ActiveMQMapMessageMarshaller . . . . .	378
6.41.3	Member Function Documentation . . . . .	378
6.41.3.1	createObject . . . . .	378
6.41.3.2	getDataStructureType . . . . .	379
6.41.3.3	looseMarshal . . . . .	379
6.41.3.4	looseUnmarshal . . . . .	379
6.41.3.5	tightMarshal1 . . . . .	380
6.41.3.6	tightMarshal2 . . . . .	380
6.41.3.7	tightUnmarshal . . . . .	381
6.42	activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller	
	Class Reference . . . . .	381
6.42.1	Detailed Description . . . . .	382
6.42.2	Constructor & Destructor Documentation . . . . .	383
6.42.2.1	ActiveMQMapMessageMarshaller . . . . .	383
6.42.2.2	~ActiveMQMapMessageMarshaller . . . . .	383
6.42.3	Member Function Documentation . . . . .	383
6.42.3.1	createObject . . . . .	383
6.42.3.2	getDataStructureType . . . . .	383
6.42.3.3	looseMarshal . . . . .	383
6.42.3.4	looseUnmarshal . . . . .	384
6.42.3.5	tightMarshal1 . . . . .	384
6.42.3.6	tightMarshal2 . . . . .	385
6.42.3.7	tightUnmarshal . . . . .	385
6.43	activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller	
	Class Reference . . . . .	386
6.43.1	Detailed Description . . . . .	387
6.43.2	Constructor & Destructor Documentation . . . . .	387
6.43.2.1	ActiveMQMapMessageMarshaller . . . . .	387
6.43.2.2	~ActiveMQMapMessageMarshaller . . . . .	387
6.43.3	Member Function Documentation . . . . .	387
6.43.3.1	createObject . . . . .	387
6.43.3.2	getDataStructureType . . . . .	387
6.43.3.3	looseMarshal . . . . .	387
6.43.3.4	looseUnmarshal . . . . .	388
6.43.3.5	tightMarshal1 . . . . .	388
6.43.3.6	tightMarshal2 . . . . .	389

6.43.3.7	tightUnmarshal . . . . .	389
6.44	activemq::commands::ActiveMQMessage Class Reference . . . . .	390
6.44.1	Constructor & Destructor Documentation . . . . .	391
6.44.1.1	ActiveMQMessage . . . . .	391
6.44.1.2	~ActiveMQMessage . . . . .	391
6.44.2	Member Function Documentation . . . . .	391
6.44.2.1	clone . . . . .	391
6.44.2.2	cloneDataStructure . . . . .	391
6.44.2.3	copyDataStructure . . . . .	391
6.44.2.4	equals . . . . .	392
6.44.2.5	getDataStructureType . . . . .	392
6.44.2.6	toString . . . . .	392
6.44.3	Field Documentation . . . . .	392
6.44.3.1	ID_ACTIVEMQMESSAGE . . . . .	392
6.45	activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller Class Reference . . . . .	393
6.45.1	Detailed Description . . . . .	394
6.45.2	Constructor & Destructor Documentation . . . . .	394
6.45.2.1	ActiveMQMessageMarshaller . . . . .	394
6.45.2.2	~ActiveMQMessageMarshaller . . . . .	394
6.45.3	Member Function Documentation . . . . .	394
6.45.3.1	createObject . . . . .	394
6.45.3.2	getDataStructureType . . . . .	394
6.45.3.3	looseMarshal . . . . .	394
6.45.3.4	looseUnmarshal . . . . .	395
6.45.3.5	tightMarshal1 . . . . .	395
6.45.3.6	tightMarshal2 . . . . .	396
6.45.3.7	tightUnmarshal . . . . .	396
6.46	activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller Class Reference . . . . .	397
6.46.1	Detailed Description . . . . .	398
6.46.2	Constructor & Destructor Documentation . . . . .	398
6.46.2.1	ActiveMQMessageMarshaller . . . . .	398
6.46.2.2	~ActiveMQMessageMarshaller . . . . .	398
6.46.3	Member Function Documentation . . . . .	398
6.46.3.1	createObject . . . . .	398
6.46.3.2	getDataStructureType . . . . .	398
6.46.3.3	looseMarshal . . . . .	399
6.46.3.4	looseUnmarshal . . . . .	399
6.46.3.5	tightMarshal1 . . . . .	400
6.46.3.6	tightMarshal2 . . . . .	400
6.46.3.7	tightUnmarshal . . . . .	401
6.47	activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller Class Reference . . . . .	401
6.47.1	Detailed Description . . . . .	402
6.47.2	Constructor & Destructor Documentation . . . . .	402
6.47.2.1	ActiveMQMessageMarshaller . . . . .	402
6.47.2.2	~ActiveMQMessageMarshaller . . . . .	402
6.47.3	Member Function Documentation . . . . .	402
6.47.3.1	createObject . . . . .	402



6.47.3.2	getDataStructureType . . . . .	403
6.47.3.3	looseMarshal . . . . .	403
6.47.3.4	looseUnmarshal . . . . .	403
6.47.3.5	tightMarshal1 . . . . .	404
6.47.3.6	tightMarshal2 . . . . .	404
6.47.3.7	tightUnmarshal . . . . .	405
6.48	activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	
	Class Reference . . . . .	405
6.48.1	Detailed Description . . . . .	406
6.48.2	Constructor & Destructor Documentation . . . . .	407
6.48.2.1	ActiveMQMessageMarshaller . . . . .	407
6.48.2.2	~ActiveMQMessageMarshaller . . . . .	407
6.48.3	Member Function Documentation . . . . .	407
6.48.3.1	createObject . . . . .	407
6.48.3.2	getDataStructureType . . . . .	407
6.48.3.3	looseMarshal . . . . .	407
6.48.3.4	looseUnmarshal . . . . .	408
6.48.3.5	tightMarshal1 . . . . .	408
6.48.3.6	tightMarshal2 . . . . .	409
6.48.3.7	tightUnmarshal . . . . .	409
6.49	activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	
	Class Reference . . . . .	410
6.49.1	Detailed Description . . . . .	411
6.49.2	Constructor & Destructor Documentation . . . . .	411
6.49.2.1	ActiveMQMessageMarshaller . . . . .	411
6.49.2.2	~ActiveMQMessageMarshaller . . . . .	411
6.49.3	Member Function Documentation . . . . .	411
6.49.3.1	createObject . . . . .	411
6.49.3.2	getDataStructureType . . . . .	411
6.49.3.3	looseMarshal . . . . .	411
6.49.3.4	looseUnmarshal . . . . .	412
6.49.3.5	tightMarshal1 . . . . .	412
6.49.3.6	tightMarshal2 . . . . .	413
6.49.3.7	tightUnmarshal . . . . .	413
6.50	activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller	
	Class Reference . . . . .	414
6.50.1	Detailed Description . . . . .	415
6.50.2	Constructor & Destructor Documentation . . . . .	415
6.50.2.1	ActiveMQMessageMarshaller . . . . .	415
6.50.2.2	~ActiveMQMessageMarshaller . . . . .	415
6.50.3	Member Function Documentation . . . . .	415
6.50.3.1	createObject . . . . .	415
6.50.3.2	getDataStructureType . . . . .	415
6.50.3.3	looseMarshal . . . . .	416
6.50.3.4	looseUnmarshal . . . . .	416
6.50.3.5	tightMarshal1 . . . . .	417
6.50.3.6	tightMarshal2 . . . . .	417
6.50.3.7	tightUnmarshal . . . . .	418
6.51	activemq::commands::ActiveMQMessageTemplate< T > Class Template Reference . . . . .	418

6.51.1	Constructor & Destructor Documentation . . . . .	422
6.51.1.1	ActiveMQMessageTemplate . . . . .	422
6.51.1.2	~ActiveMQMessageTemplate . . . . .	422
6.51.2	Member Function Documentation . . . . .	422
6.51.2.1	acknowledge . . . . .	422
6.51.2.2	clearBody . . . . .	422
6.51.2.3	clearProperties . . . . .	423
6.51.2.4	equals . . . . .	423
6.51.2.5	failIfReadOnlyBody . . . . .	423
6.51.2.6	failIfReadOnlyProperties . . . . .	423
6.51.2.7	failIfWriteOnlyBody . . . . .	423
6.51.2.8	getBooleanProperty . . . . .	423
6.51.2.9	getByteProperty . . . . .	424
6.51.2.10	getCMSCorrelationID . . . . .	424
6.51.2.11	getCMSDeliveryMode . . . . .	425
6.51.2.12	getCMSDestination . . . . .	425
6.51.2.13	getCMSExpiration . . . . .	425
6.51.2.14	getCMSMessageID . . . . .	426
6.51.2.15	getCMSPriority . . . . .	426
6.51.2.16	getCMSRedelivered . . . . .	426
6.51.2.17	getCMSReplyTo . . . . .	427
6.51.2.18	getCMSTimestamp . . . . .	427
6.51.2.19	getCMSType . . . . .	427
6.51.2.20	getDoubleProperty . . . . .	428
6.51.2.21	getFloatProperty . . . . .	428
6.51.2.22	getIntProperty . . . . .	429
6.51.2.23	getLongProperty . . . . .	429
6.51.2.24	getPropertyNames . . . . .	430
6.51.2.25	getShortProperty . . . . .	430
6.51.2.26	getStringProperty . . . . .	430
6.51.2.27	onSend . . . . .	431
6.51.2.28	propertyExists . . . . .	431
6.51.2.29	setBooleanProperty . . . . .	431
6.51.2.30	setByteProperty . . . . .	432
6.51.2.31	setCMSCorrelationID . . . . .	432
6.51.2.32	setCMSDeliveryMode . . . . .	432
6.51.2.33	setCMSDestination . . . . .	433
6.51.2.34	setCMSExpiration . . . . .	433
6.51.2.35	setCMSMessageID . . . . .	433
6.51.2.36	setCMSPriority . . . . .	434
6.51.2.37	setCMSRedelivered . . . . .	434
6.51.2.38	setCMSReplyTo . . . . .	434
6.51.2.39	setCMSTimestamp . . . . .	435
6.51.2.40	setCMSType . . . . .	435
6.51.2.41	setDoubleProperty . . . . .	435
6.51.2.42	setFloatProperty . . . . .	436
6.51.2.43	setIntProperty . . . . .	436
6.51.2.44	setLongProperty . . . . .	436
6.51.2.45	setShortProperty . . . . .	437
6.51.2.46	setStringProperty . . . . .	437

6.52	activemq::commands::ActiveMQObjectMessage Class Reference . . .	438
6.52.1	Constructor & Destructor Documentation . . . . .	439
6.52.1.1	ActiveMQObjectMessage . . . . .	439
6.52.1.2	~ActiveMQObjectMessage . . . . .	439
6.52.2	Member Function Documentation . . . . .	439
6.52.2.1	clone . . . . .	439
6.52.2.2	cloneDataStructure . . . . .	439
6.52.2.3	copyDataStructure . . . . .	439
6.52.2.4	equals . . . . .	439
6.52.2.5	getDataStructureType . . . . .	440
6.52.2.6	toString . . . . .	440
6.52.3	Field Documentation . . . . .	440
6.52.3.1	ID_ACTIVEMQOBJECTMESSAGE . . . . .	440
6.53	activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller Class Reference . . . . .	440
6.53.1	Detailed Description . . . . .	441
6.53.2	Constructor & Destructor Documentation . . . . .	442
6.53.2.1	ActiveMQObjectMessageMarshaller . . . . .	442
6.53.2.2	~ActiveMQObjectMessageMarshaller . . . . .	442
6.53.3	Member Function Documentation . . . . .	442
6.53.3.1	createObject . . . . .	442
6.53.3.2	getDataStructureType . . . . .	442
6.53.3.3	looseMarshal . . . . .	442
6.53.3.4	looseUnmarshal . . . . .	443
6.53.3.5	tightMarshal1 . . . . .	443
6.53.3.6	tightMarshal2 . . . . .	444
6.53.3.7	tightUnmarshal . . . . .	444
6.54	activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller Class Reference . . . . .	445
6.54.1	Detailed Description . . . . .	446
6.54.2	Constructor & Destructor Documentation . . . . .	446
6.54.2.1	ActiveMQObjectMessageMarshaller . . . . .	446
6.54.2.2	~ActiveMQObjectMessageMarshaller . . . . .	446
6.54.3	Member Function Documentation . . . . .	446
6.54.3.1	createObject . . . . .	446
6.54.3.2	getDataStructureType . . . . .	446
6.54.3.3	looseMarshal . . . . .	446
6.54.3.4	looseUnmarshal . . . . .	447
6.54.3.5	tightMarshal1 . . . . .	447
6.54.3.6	tightMarshal2 . . . . .	448
6.54.3.7	tightUnmarshal . . . . .	448
6.55	activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller Class Reference . . . . .	449
6.55.1	Detailed Description . . . . .	450
6.55.2	Constructor & Destructor Documentation . . . . .	450
6.55.2.1	ActiveMQObjectMessageMarshaller . . . . .	450
6.55.2.2	~ActiveMQObjectMessageMarshaller . . . . .	450
6.55.3	Member Function Documentation . . . . .	450
6.55.3.1	createObject . . . . .	450
6.55.3.2	getDataStructureType . . . . .	450

6.55.3.3	looseMarshal . . . . .	451
6.55.3.4	looseUnmarshal . . . . .	451
6.55.3.5	tightMarshal1 . . . . .	452
6.55.3.6	tightMarshal2 . . . . .	452
6.55.3.7	tightUnmarshal . . . . .	453
6.56	activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	
	Class Reference . . . . .	453
6.56.1	Detailed Description . . . . .	454
6.56.2	Constructor & Destructor Documentation . . . . .	454
6.56.2.1	ActiveMQObjectMessageMarshaller . . . . .	454
6.56.2.2	~ActiveMQObjectMessageMarshaller . . . . .	454
6.56.3	Member Function Documentation . . . . .	454
6.56.3.1	createObject . . . . .	454
6.56.3.2	getDataStructureType . . . . .	455
6.56.3.3	looseMarshal . . . . .	455
6.56.3.4	looseUnmarshal . . . . .	455
6.56.3.5	tightMarshal1 . . . . .	456
6.56.3.6	tightMarshal2 . . . . .	456
6.56.3.7	tightUnmarshal . . . . .	457
6.57	activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller	
	Class Reference . . . . .	457
6.57.1	Detailed Description . . . . .	458
6.57.2	Constructor & Destructor Documentation . . . . .	459
6.57.2.1	ActiveMQObjectMessageMarshaller . . . . .	459
6.57.2.2	~ActiveMQObjectMessageMarshaller . . . . .	459
6.57.3	Member Function Documentation . . . . .	459
6.57.3.1	createObject . . . . .	459
6.57.3.2	getDataStructureType . . . . .	459
6.57.3.3	looseMarshal . . . . .	459
6.57.3.4	looseUnmarshal . . . . .	460
6.57.3.5	tightMarshal1 . . . . .	460
6.57.3.6	tightMarshal2 . . . . .	461
6.57.3.7	tightUnmarshal . . . . .	461
6.58	activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller	
	Class Reference . . . . .	462
6.58.1	Detailed Description . . . . .	463
6.58.2	Constructor & Destructor Documentation . . . . .	463
6.58.2.1	ActiveMQObjectMessageMarshaller . . . . .	463
6.58.2.2	~ActiveMQObjectMessageMarshaller . . . . .	463
6.58.3	Member Function Documentation . . . . .	463
6.58.3.1	createObject . . . . .	463
6.58.3.2	getDataStructureType . . . . .	463
6.58.3.3	looseMarshal . . . . .	463
6.58.3.4	looseUnmarshal . . . . .	464
6.58.3.5	tightMarshal1 . . . . .	464
6.58.3.6	tightMarshal2 . . . . .	465
6.58.3.7	tightUnmarshal . . . . .	465
6.59	activemq::core::ActiveMQProducer Class Reference . . . . .	466
6.59.1	Constructor & Destructor Documentation . . . . .	468
6.59.1.1	ActiveMQProducer . . . . .	468

6.59.1.2	~ActiveMQProducer . . . . .	468
6.59.2	Member Function Documentation . . . . .	468
6.59.2.1	close . . . . .	468
6.59.2.2	getDeliveryMode . . . . .	469
6.59.2.3	getDisableMessageID . . . . .	469
6.59.2.4	getDisableMessageTimeStamp . . . . .	469
6.59.2.5	getPriority . . . . .	469
6.59.2.6	getProducerId . . . . .	469
6.59.2.7	getProducerInfo . . . . .	470
6.59.2.8	getSendTimeout . . . . .	470
6.59.2.9	getTimeToLive . . . . .	470
6.59.2.10	isClosed . . . . .	470
6.59.2.11	onProducerAck . . . . .	470
6.59.2.12	send . . . . .	471
6.59.2.13	send . . . . .	471
6.59.2.14	send . . . . .	472
6.59.2.15	send . . . . .	472
6.59.2.16	setDeliveryMode . . . . .	473
6.59.2.17	setDisableMessageID . . . . .	473
6.59.2.18	setDisableMessageTimeStamp . . . . .	473
6.59.2.19	setPriority . . . . .	474
6.59.2.20	setSendTimeout . . . . .	474
6.59.2.21	setTimeToLive . . . . .	474
6.60	activemq::util::ActiveMQProperties Class Reference . . . . .	474
6.60.1	Detailed Description . . . . .	476
6.60.2	Constructor & Destructor Documentation . . . . .	476
6.60.2.1	ActiveMQProperties . . . . .	476
6.60.2.2	~ActiveMQProperties . . . . .	476
6.60.3	Member Function Documentation . . . . .	476
6.60.3.1	clear . . . . .	476
6.60.3.2	clone . . . . .	476
6.60.3.3	copy . . . . .	476
6.60.3.4	getProperties . . . . .	477
6.60.3.5	getProperties . . . . .	477
6.60.3.6	getProperty . . . . .	477
6.60.3.7	getProperty . . . . .	477
6.60.3.8	hasProperty . . . . .	477
6.60.3.9	isEmpty . . . . .	478
6.60.3.10	remove . . . . .	478
6.60.3.11	setProperties . . . . .	478
6.60.3.12	setProperty . . . . .	478
6.60.3.13	toArray . . . . .	479
6.60.3.14	toString . . . . .	479
6.61	activemq::commands::ActiveMQQueue Class Reference . . . . .	479
6.61.1	Constructor & Destructor Documentation . . . . .	480
6.61.1.1	ActiveMQQueue . . . . .	480
6.61.1.2	ActiveMQQueue . . . . .	480
6.61.1.3	~ActiveMQQueue . . . . .	480
6.61.2	Member Function Documentation . . . . .	480
6.61.2.1	clone . . . . .	480

6.61.2.2	cloneDataStructure . . . . .	481
6.61.2.3	copy . . . . .	481
6.61.2.4	copyDataStructure . . . . .	481
6.61.2.5	equals . . . . .	481
6.61.2.6	getCMSDestination . . . . .	482
6.61.2.7	getCMSProperties . . . . .	482
6.61.2.8	getDataStructureType . . . . .	482
6.61.2.9	getDestinationType . . . . .	482
6.61.2.10	getQueueName . . . . .	483
6.61.2.11	toString . . . . .	483
6.61.3	Field Documentation . . . . .	483
6.61.3.1	ID_ACTIVEMQQUEUE . . . . .	483
6.62	activemq::core::ActiveMQQueueBrowser Class Reference . . . . .	483
6.62.1	Constructor & Destructor Documentation . . . . .	484
6.62.1.1	ActiveMQQueueBrowser . . . . .	484
6.62.1.2	~ActiveMQQueueBrowser . . . . .	484
6.62.2	Member Function Documentation . . . . .	484
6.62.2.1	close . . . . .	484
6.62.2.2	getEnumeration . . . . .	485
6.62.2.3	getMessageSelector . . . . .	485
6.62.2.4	getQueue . . . . .	485
6.62.2.5	hasMoreMessages . . . . .	486
6.62.2.6	nextMessage . . . . .	486
6.62.3	Friends And Related Function Documentation . . . . .	486
6.62.3.1	Browser . . . . .	486
6.63	activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller Class Reference . . . . .	487
6.63.1	Detailed Description . . . . .	488
6.63.2	Constructor & Destructor Documentation . . . . .	488
6.63.2.1	ActiveMQQueueMarshaller . . . . .	488
6.63.2.2	~ActiveMQQueueMarshaller . . . . .	488
6.63.3	Member Function Documentation . . . . .	488
6.63.3.1	createObject . . . . .	488
6.63.3.2	getDataStructureType . . . . .	488
6.63.3.3	looseMarshal . . . . .	488
6.63.3.4	looseUnmarshal . . . . .	489
6.63.3.5	tightMarshal1 . . . . .	489
6.63.3.6	tightMarshal2 . . . . .	490
6.63.3.7	tightUnmarshal . . . . .	490
6.64	activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller Class Reference . . . . .	491
6.64.1	Detailed Description . . . . .	492
6.64.2	Constructor & Destructor Documentation . . . . .	492
6.64.2.1	ActiveMQQueueMarshaller . . . . .	492
6.64.2.2	~ActiveMQQueueMarshaller . . . . .	492
6.64.3	Member Function Documentation . . . . .	492
6.64.3.1	createObject . . . . .	492
6.64.3.2	getDataStructureType . . . . .	492
6.64.3.3	looseMarshal . . . . .	493
6.64.3.4	looseUnmarshal . . . . .	493

6.64.3.5	tightMarshal1	494
6.64.3.6	tightMarshal2	494
6.64.3.7	tightUnmarshal	495
6.65	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller	
	Class Reference	495
6.65.1	Detailed Description	496
6.65.2	Constructor & Destructor Documentation	496
6.65.2.1	ActiveMQQueueMarshaller	496
6.65.2.2	~ActiveMQQueueMarshaller	496
6.65.3	Member Function Documentation	496
6.65.3.1	createObject	496
6.65.3.2	getDataStructureType	497
6.65.3.3	looseMarshal	497
6.65.3.4	looseUnmarshal	497
6.65.3.5	tightMarshal1	498
6.65.3.6	tightMarshal2	498
6.65.3.7	tightUnmarshal	499
6.66	activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller	
	Class Reference	499
6.66.1	Detailed Description	500
6.66.2	Constructor & Destructor Documentation	501
6.66.2.1	ActiveMQQueueMarshaller	501
6.66.2.2	~ActiveMQQueueMarshaller	501
6.66.3	Member Function Documentation	501
6.66.3.1	createObject	501
6.66.3.2	getDataStructureType	501
6.66.3.3	looseMarshal	501
6.66.3.4	looseUnmarshal	502
6.66.3.5	tightMarshal1	502
6.66.3.6	tightMarshal2	503
6.66.3.7	tightUnmarshal	503
6.67	activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	
	Class Reference	504
6.67.1	Detailed Description	505
6.67.2	Constructor & Destructor Documentation	505
6.67.2.1	ActiveMQQueueMarshaller	505
6.67.2.2	~ActiveMQQueueMarshaller	505
6.67.3	Member Function Documentation	505
6.67.3.1	createObject	505
6.67.3.2	getDataStructureType	505
6.67.3.3	looseMarshal	505
6.67.3.4	looseUnmarshal	506
6.67.3.5	tightMarshal1	506
6.67.3.6	tightMarshal2	507
6.67.3.7	tightUnmarshal	507
6.68	activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller	
	Class Reference	508
6.68.1	Detailed Description	509
6.68.2	Constructor & Destructor Documentation	509
6.68.2.1	ActiveMQQueueMarshaller	509

6.68.2.2	~ActiveMQQueueMarshaller . . . . .	509
6.68.3	Member Function Documentation . . . . .	509
6.68.3.1	createObject . . . . .	509
6.68.3.2	getDataStructureType . . . . .	509
6.68.3.3	looseMarshal . . . . .	510
6.68.3.4	looseUnmarshal . . . . .	510
6.68.3.5	tightMarshal1 . . . . .	511
6.68.3.6	tightMarshal2 . . . . .	511
6.68.3.7	tightUnmarshal . . . . .	512
6.69	activemq::core::ActiveMQSession Class Reference . . . . .	512
6.69.1	Constructor & Destructor Documentation . . . . .	517
6.69.1.1	ActiveMQSession . . . . .	517
6.69.1.2	~ActiveMQSession . . . . .	517
6.69.2	Member Function Documentation . . . . .	517
6.69.2.1	acknowledge . . . . .	517
6.69.2.2	addConsumer . . . . .	517
6.69.2.3	addProducer . . . . .	517
6.69.2.4	clearMessagesInProgress . . . . .	518
6.69.2.5	close . . . . .	518
6.69.2.6	commit . . . . .	518
6.69.2.7	createBrowser . . . . .	518
6.69.2.8	createBrowser . . . . .	519
6.69.2.9	createBytesMessage . . . . .	519
6.69.2.10	createBytesMessage . . . . .	520
6.69.2.11	createConsumer . . . . .	520
6.69.2.12	createConsumer . . . . .	520
6.69.2.13	createConsumer . . . . .	521
6.69.2.14	createDurableConsumer . . . . .	521
6.69.2.15	createMapMessage . . . . .	522
6.69.2.16	createMessage . . . . .	522
6.69.2.17	createProducer . . . . .	522
6.69.2.18	createQueue . . . . .	522
6.69.2.19	createStreamMessage . . . . .	523
6.69.2.20	createTemporaryQueue . . . . .	523
6.69.2.21	createTemporaryTopic . . . . .	523
6.69.2.22	createTextMessage . . . . .	524
6.69.2.23	createTextMessage . . . . .	524
6.69.2.24	createTopic . . . . .	524
6.69.2.25	deliverAcks . . . . .	525
6.69.2.26	dispatch . . . . .	525
6.69.2.27	doStartTransaction . . . . .	525
6.69.2.28	fire . . . . .	525
6.69.2.29	getAcknowledgeMode . . . . .	525
6.69.2.30	getConnection . . . . .	526
6.69.2.31	getExceptionListener . . . . .	526
6.69.2.32	getLastDeliveredSequenceId . . . . .	526
6.69.2.33	getNextConsumerId . . . . .	526
6.69.2.34	getNextProducerId . . . . .	526
6.69.2.35	getSessionId . . . . .	527
6.69.2.36	getSessionInfo . . . . .	527



6.69.2.37	getTransactionContext . . . . .	527
6.69.2.38	isAutoAcknowledge . . . . .	527
6.69.2.39	isClientAcknowledge . . . . .	527
6.69.2.40	isDupsOkAcknowledge . . . . .	527
6.69.2.41	isIndividualAcknowledge . . . . .	528
6.69.2.42	isStarted . . . . .	528
6.69.2.43	isTransacted . . . . .	528
6.69.2.44	oneway . . . . .	528
6.69.2.45	recover . . . . .	528
6.69.2.46	redispatch . . . . .	529
6.69.2.47	removeConsumer . . . . .	529
6.69.2.48	removeProducer . . . . .	529
6.69.2.49	rollback . . . . .	530
6.69.2.50	send . . . . .	530
6.69.2.51	setLastDeliveredSequenceId . . . . .	530
6.69.2.52	start . . . . .	531
6.69.2.53	stop . . . . .	531
6.69.2.54	syncRequest . . . . .	531
6.69.2.55	unsubscribe . . . . .	531
6.69.2.56	wakeup . . . . .	532
6.69.3	Friends And Related Function Documentation . . . . .	532
6.69.3.1	ActiveMQSessionExecutor . . . . .	532
6.70	activemq::core::ActiveMQSessionExecutor Class Reference . . . . .	532
6.70.1	Detailed Description . . . . .	533
6.70.2	Constructor & Destructor Documentation . . . . .	533
6.70.2.1	ActiveMQSessionExecutor . . . . .	533
6.70.2.2	~ActiveMQSessionExecutor . . . . .	533
6.70.3	Member Function Documentation . . . . .	533
6.70.3.1	clear . . . . .	533
6.70.3.2	clearMessagesInProgress . . . . .	533
6.70.3.3	close . . . . .	534
6.70.3.4	execute . . . . .	534
6.70.3.5	executeFirst . . . . .	534
6.70.3.6	getUnconsumedMessages . . . . .	534
6.70.3.7	hasUnconsumedMessages . . . . .	534
6.70.3.8	isEmpty . . . . .	535
6.70.3.9	isRunning . . . . .	535
6.70.3.10	iterate . . . . .	535
6.70.3.11	start . . . . .	535
6.70.3.12	stop . . . . .	535
6.70.3.13	wakeup . . . . .	535
6.71	activemq::commands::ActiveMQStreamMessage Class Reference . . . . .	535
6.71.1	Constructor & Destructor Documentation . . . . .	539
6.71.1.1	ActiveMQStreamMessage . . . . .	539
6.71.1.2	~ActiveMQStreamMessage . . . . .	539
6.71.2	Member Function Documentation . . . . .	539
6.71.2.1	clearBody . . . . .	539
6.71.2.2	clone . . . . .	539
6.71.2.3	cloneDataStructure . . . . .	539
6.71.2.4	copyDataStructure . . . . .	540

6.71.2.5	equals	540
6.71.2.6	getDataStructureType	540
6.71.2.7	onSend	540
6.71.2.8	readBoolean	541
6.71.2.9	readByte	541
6.71.2.10	readBytes	542
6.71.2.11	readBytes	542
6.71.2.12	readChar	543
6.71.2.13	readDouble	544
6.71.2.14	readFloat	544
6.71.2.15	readInt	545
6.71.2.16	readLong	545
6.71.2.17	readShort	546
6.71.2.18	readString	546
6.71.2.19	readUnsignedShort	547
6.71.2.20	reset	547
6.71.2.21	toString	547
6.71.2.22	writeBoolean	548
6.71.2.23	writeByte	548
6.71.2.24	writeBytes	548
6.71.2.25	writeBytes	549
6.71.2.26	writeChar	549
6.71.2.27	writeDouble	550
6.71.2.28	writeFloat	550
6.71.2.29	writeInt	551
6.71.2.30	writeLong	551
6.71.2.31	writeShort	551
6.71.2.32	writeString	552
6.71.2.33	writeUnsignedShort	552
6.71.3	Field Documentation	553
6.71.3.1	ID_ACTIVEMQSTREAMMESSAGE	553
6.72	activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller	
	Class Reference	553
6.72.1	Detailed Description	554
6.72.2	Constructor & Destructor Documentation	554
6.72.2.1	ActiveMQStreamMessageMarshaller	554
6.72.2.2	~ActiveMQStreamMessageMarshaller	554
6.72.3	Member Function Documentation	554
6.72.3.1	createObject	554
6.72.3.2	getDataStructureType	554
6.72.3.3	looseMarshal	555
6.72.3.4	looseUnmarshal	555
6.72.3.5	tightMarshal1	556
6.72.3.6	tightMarshal2	556
6.72.3.7	tightUnmarshal	557
6.73	activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller	
	Class Reference	557
6.73.1	Detailed Description	558
6.73.2	Constructor & Destructor Documentation	558
6.73.2.1	ActiveMQStreamMessageMarshaller	558

6.73.2.2	~ActiveMQStreamMessageMarshaller . . . . .	558
6.73.3	Member Function Documentation . . . . .	558
6.73.3.1	createObject . . . . .	558
6.73.3.2	getDataStructureType . . . . .	559
6.73.3.3	looseMarshal . . . . .	559
6.73.3.4	looseUnmarshal . . . . .	559
6.73.3.5	tightMarshal1 . . . . .	560
6.73.3.6	tightMarshal2 . . . . .	560
6.73.3.7	tightUnmarshal . . . . .	561
6.74	activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller	
	Class Reference . . . . .	561
6.74.1	Detailed Description . . . . .	562
6.74.2	Constructor & Destructor Documentation . . . . .	563
6.74.2.1	ActiveMQStreamMessageMarshaller . . . . .	563
6.74.2.2	~ActiveMQStreamMessageMarshaller . . . . .	563
6.74.3	Member Function Documentation . . . . .	563
6.74.3.1	createObject . . . . .	563
6.74.3.2	getDataStructureType . . . . .	563
6.74.3.3	looseMarshal . . . . .	563
6.74.3.4	looseUnmarshal . . . . .	564
6.74.3.5	tightMarshal1 . . . . .	564
6.74.3.6	tightMarshal2 . . . . .	565
6.74.3.7	tightUnmarshal . . . . .	565
6.75	activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller	
	Class Reference . . . . .	566
6.75.1	Detailed Description . . . . .	567
6.75.2	Constructor & Destructor Documentation . . . . .	567
6.75.2.1	ActiveMQStreamMessageMarshaller . . . . .	567
6.75.2.2	~ActiveMQStreamMessageMarshaller . . . . .	567
6.75.3	Member Function Documentation . . . . .	567
6.75.3.1	createObject . . . . .	567
6.75.3.2	getDataStructureType . . . . .	567
6.75.3.3	looseMarshal . . . . .	567
6.75.3.4	looseUnmarshal . . . . .	568
6.75.3.5	tightMarshal1 . . . . .	568
6.75.3.6	tightMarshal2 . . . . .	569
6.75.3.7	tightUnmarshal . . . . .	569
6.76	activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller	
	Class Reference . . . . .	570
6.76.1	Detailed Description . . . . .	571
6.76.2	Constructor & Destructor Documentation . . . . .	571
6.76.2.1	ActiveMQStreamMessageMarshaller . . . . .	571
6.76.2.2	~ActiveMQStreamMessageMarshaller . . . . .	571
6.76.3	Member Function Documentation . . . . .	571
6.76.3.1	createObject . . . . .	571
6.76.3.2	getDataStructureType . . . . .	571
6.76.3.3	looseMarshal . . . . .	572
6.76.3.4	looseUnmarshal . . . . .	572
6.76.3.5	tightMarshal1 . . . . .	573
6.76.3.6	tightMarshal2 . . . . .	573

6.76.3.7	tightUnmarshal . . . . .	574
6.77	activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller	
	Class Reference . . . . .	574
6.77.1	Detailed Description . . . . .	575
6.77.2	Constructor & Destructor Documentation . . . . .	575
6.77.2.1	ActiveMQStreamMessageMarshaller . . . . .	575
6.77.2.2	~ActiveMQStreamMessageMarshaller . . . . .	575
6.77.3	Member Function Documentation . . . . .	575
6.77.3.1	createObject . . . . .	575
6.77.3.2	getDataStructureType . . . . .	576
6.77.3.3	looseMarshal . . . . .	576
6.77.3.4	looseUnmarshal . . . . .	576
6.77.3.5	tightMarshal1 . . . . .	577
6.77.3.6	tightMarshal2 . . . . .	577
6.77.3.7	tightUnmarshal . . . . .	578
6.78	activemq::commands::ActiveMQTempDestination Class Reference . .	578
6.78.1	Constructor & Destructor Documentation . . . . .	580
6.78.1.1	ActiveMQTempDestination . . . . .	580
6.78.1.2	ActiveMQTempDestination . . . . .	580
6.78.1.3	~ActiveMQTempDestination . . . . .	580
6.78.2	Member Function Documentation . . . . .	580
6.78.2.1	cloneDataStructure . . . . .	580
6.78.2.2	close . . . . .	580
6.78.2.3	copyDataStructure . . . . .	580
6.78.2.4	equals . . . . .	581
6.78.2.5	getDataStructureType . . . . .	581
6.78.2.6	setConnection . . . . .	581
6.78.2.7	toString . . . . .	582
6.78.3	Field Documentation . . . . .	582
6.78.3.1	connection . . . . .	582
6.78.3.2	ID_ACTIVEMQTEMPDESTINATION . . . . .	582
6.79	activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller	
	Class Reference . . . . .	582
6.79.1	Detailed Description . . . . .	583
6.79.2	Constructor & Destructor Documentation . . . . .	584
6.79.2.1	ActiveMQTempDestinationMarshaller . . . . .	584
6.79.2.2	~ActiveMQTempDestinationMarshaller . . . . .	584
6.79.3	Member Function Documentation . . . . .	584
6.79.3.1	looseMarshal . . . . .	584
6.79.3.2	looseUnmarshal . . . . .	584
6.79.3.3	tightMarshal1 . . . . .	585
6.79.3.4	tightMarshal2 . . . . .	585
6.79.3.5	tightUnmarshal . . . . .	586
6.80	activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller	
	Class Reference . . . . .	587
6.80.1	Detailed Description . . . . .	587
6.80.2	Constructor & Destructor Documentation . . . . .	588
6.80.2.1	ActiveMQTempDestinationMarshaller . . . . .	588
6.80.2.2	~ActiveMQTempDestinationMarshaller . . . . .	588
6.80.3	Member Function Documentation . . . . .	588

6.80.3.1	looseMarshal . . . . .	588
6.80.3.2	looseUnmarshal . . . . .	588
6.80.3.3	tightMarshal1 . . . . .	589
6.80.3.4	tightMarshal2 . . . . .	589
6.80.3.5	tightUnmarshal . . . . .	590
6.81	activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller	
	Class Reference . . . . .	591
6.81.1	Detailed Description . . . . .	591
6.81.2	Constructor & Destructor Documentation . . . . .	592
6.81.2.1	ActiveMQTempDestinationMarshaller . . . . .	592
6.81.2.2	~ActiveMQTempDestinationMarshaller . . . . .	592
6.81.3	Member Function Documentation . . . . .	592
6.81.3.1	looseMarshal . . . . .	592
6.81.3.2	looseUnmarshal . . . . .	592
6.81.3.3	tightMarshal1 . . . . .	593
6.81.3.4	tightMarshal2 . . . . .	593
6.81.3.5	tightUnmarshal . . . . .	594
6.82	activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller	
	Class Reference . . . . .	595
6.82.1	Detailed Description . . . . .	595
6.82.2	Constructor & Destructor Documentation . . . . .	596
6.82.2.1	ActiveMQTempDestinationMarshaller . . . . .	596
6.82.2.2	~ActiveMQTempDestinationMarshaller . . . . .	596
6.82.3	Member Function Documentation . . . . .	596
6.82.3.1	looseMarshal . . . . .	596
6.82.3.2	looseUnmarshal . . . . .	596
6.82.3.3	tightMarshal1 . . . . .	597
6.82.3.4	tightMarshal2 . . . . .	597
6.82.3.5	tightUnmarshal . . . . .	598
6.83	activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller	
	Class Reference . . . . .	599
6.83.1	Detailed Description . . . . .	599
6.83.2	Constructor & Destructor Documentation . . . . .	600
6.83.2.1	ActiveMQTempDestinationMarshaller . . . . .	600
6.83.2.2	~ActiveMQTempDestinationMarshaller . . . . .	600
6.83.3	Member Function Documentation . . . . .	600
6.83.3.1	looseMarshal . . . . .	600
6.83.3.2	looseUnmarshal . . . . .	600
6.83.3.3	tightMarshal1 . . . . .	601
6.83.3.4	tightMarshal2 . . . . .	601
6.83.3.5	tightUnmarshal . . . . .	602
6.84	activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller	
	Class Reference . . . . .	603
6.84.1	Detailed Description . . . . .	603
6.84.2	Constructor & Destructor Documentation . . . . .	604
6.84.2.1	ActiveMQTempDestinationMarshaller . . . . .	604
6.84.2.2	~ActiveMQTempDestinationMarshaller . . . . .	604
6.84.3	Member Function Documentation . . . . .	604
6.84.3.1	looseMarshal . . . . .	604
6.84.3.2	looseUnmarshal . . . . .	604

6.84.3.3	tightMarshal1	605
6.84.3.4	tightMarshal2	605
6.84.3.5	tightUnmarshal	606
6.85	activemq::commands::ActiveMQTempQueue Class Reference	606
6.85.1	Constructor & Destructor Documentation	608
6.85.1.1	ActiveMQTempQueue	608
6.85.1.2	ActiveMQTempQueue	608
6.85.1.3	~ActiveMQTempQueue	608
6.85.2	Member Function Documentation	608
6.85.2.1	clone	608
6.85.2.2	cloneDataStructure	608
6.85.2.3	copy	608
6.85.2.4	copyDataStructure	609
6.85.2.5	destroy	609
6.85.2.6	equals	609
6.85.2.7	getCMSDestination	609
6.85.2.8	getCMSProperties	610
6.85.2.9	getDataStructureType	610
6.85.2.10	getDestinationType	610
6.85.2.11	getQueueName	610
6.85.2.12	toString	611
6.85.3	Field Documentation	611
6.85.3.1	ID_ACTIVEMQTEMPQUEUE	611
6.86	activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller Class Reference	611
6.86.1	Detailed Description	612
6.86.2	Constructor & Destructor Documentation	612
6.86.2.1	ActiveMQTempQueueMarshaller	612
6.86.2.2	~ActiveMQTempQueueMarshaller	612
6.86.3	Member Function Documentation	612
6.86.3.1	createObject	612
6.86.3.2	getDataStructureType	613
6.86.3.3	looseMarshal	613
6.86.3.4	looseUnmarshal	613
6.86.3.5	tightMarshal1	614
6.86.3.6	tightMarshal2	614
6.86.3.7	tightUnmarshal	615
6.87	activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller Class Reference	615
6.87.1	Detailed Description	616
6.87.2	Constructor & Destructor Documentation	616
6.87.2.1	ActiveMQTempQueueMarshaller	616
6.87.2.2	~ActiveMQTempQueueMarshaller	616
6.87.3	Member Function Documentation	616
6.87.3.1	createObject	616
6.87.3.2	getDataStructureType	617
6.87.3.3	looseMarshal	617
6.87.3.4	looseUnmarshal	617
6.87.3.5	tightMarshal1	618
6.87.3.6	tightMarshal2	618

6.87.3.7	tightUnmarshal . . . . .	619
6.88	activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	
	Class Reference . . . . .	619
6.88.1	Detailed Description . . . . .	620
6.88.2	Constructor & Destructor Documentation . . . . .	621
6.88.2.1	ActiveMQTempQueueMarshaller . . . . .	621
6.88.2.2	~ActiveMQTempQueueMarshaller . . . . .	621
6.88.3	Member Function Documentation . . . . .	621
6.88.3.1	createObject . . . . .	621
6.88.3.2	getDataStructureType . . . . .	621
6.88.3.3	looseMarshal . . . . .	621
6.88.3.4	looseUnmarshal . . . . .	622
6.88.3.5	tightMarshal1 . . . . .	622
6.88.3.6	tightMarshal2 . . . . .	623
6.88.3.7	tightUnmarshal . . . . .	623
6.89	activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	
	Class Reference . . . . .	624
6.89.1	Detailed Description . . . . .	625
6.89.2	Constructor & Destructor Documentation . . . . .	625
6.89.2.1	ActiveMQTempQueueMarshaller . . . . .	625
6.89.2.2	~ActiveMQTempQueueMarshaller . . . . .	625
6.89.3	Member Function Documentation . . . . .	625
6.89.3.1	createObject . . . . .	625
6.89.3.2	getDataStructureType . . . . .	625
6.89.3.3	looseMarshal . . . . .	625
6.89.3.4	looseUnmarshal . . . . .	626
6.89.3.5	tightMarshal1 . . . . .	626
6.89.3.6	tightMarshal2 . . . . .	627
6.89.3.7	tightUnmarshal . . . . .	627
6.90	activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller	
	Class Reference . . . . .	628
6.90.1	Detailed Description . . . . .	629
6.90.2	Constructor & Destructor Documentation . . . . .	629
6.90.2.1	ActiveMQTempQueueMarshaller . . . . .	629
6.90.2.2	~ActiveMQTempQueueMarshaller . . . . .	629
6.90.3	Member Function Documentation . . . . .	629
6.90.3.1	createObject . . . . .	629
6.90.3.2	getDataStructureType . . . . .	629
6.90.3.3	looseMarshal . . . . .	630
6.90.3.4	looseUnmarshal . . . . .	630
6.90.3.5	tightMarshal1 . . . . .	631
6.90.3.6	tightMarshal2 . . . . .	631
6.90.3.7	tightUnmarshal . . . . .	632
6.91	activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller	
	Class Reference . . . . .	632
6.91.1	Detailed Description . . . . .	633
6.91.2	Constructor & Destructor Documentation . . . . .	633
6.91.2.1	ActiveMQTempQueueMarshaller . . . . .	633
6.91.2.2	~ActiveMQTempQueueMarshaller . . . . .	633
6.91.3	Member Function Documentation . . . . .	633

6.91.3.1	createObject . . . . .	633
6.91.3.2	getDataStructureType . . . . .	634
6.91.3.3	looseMarshal . . . . .	634
6.91.3.4	looseUnmarshal . . . . .	634
6.91.3.5	tightMarshal1 . . . . .	635
6.91.3.6	tightMarshal2 . . . . .	635
6.91.3.7	tightUnmarshal . . . . .	636
6.92	activemq::commands::ActiveMQTempTopic Class Reference . . . . .	636
6.92.1	Constructor & Destructor Documentation . . . . .	638
6.92.1.1	ActiveMQTempTopic . . . . .	638
6.92.1.2	ActiveMQTempTopic . . . . .	638
6.92.1.3	~ActiveMQTempTopic . . . . .	638
6.92.2	Member Function Documentation . . . . .	638
6.92.2.1	clone . . . . .	638
6.92.2.2	cloneDataStructure . . . . .	638
6.92.2.3	copy . . . . .	638
6.92.2.4	copyDataStructure . . . . .	639
6.92.2.5	destroy . . . . .	639
6.92.2.6	equals . . . . .	639
6.92.2.7	getCMSDestination . . . . .	639
6.92.2.8	getCMSProperties . . . . .	640
6.92.2.9	getDataStructureType . . . . .	640
6.92.2.10	getDestinationType . . . . .	640
6.92.2.11	getTopicName . . . . .	640
6.92.2.12	toString . . . . .	641
6.92.3	Field Documentation . . . . .	641
6.92.3.1	ID_ACTIVEMQTEMPTOPIC . . . . .	641
6.93	activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller Class Reference . . . . .	641
6.93.1	Detailed Description . . . . .	642
6.93.2	Constructor & Destructor Documentation . . . . .	642
6.93.2.1	ActiveMQTempTopicMarshaller . . . . .	642
6.93.2.2	~ActiveMQTempTopicMarshaller . . . . .	642
6.93.3	Member Function Documentation . . . . .	642
6.93.3.1	createObject . . . . .	642
6.93.3.2	getDataStructureType . . . . .	643
6.93.3.3	looseMarshal . . . . .	643
6.93.3.4	looseUnmarshal . . . . .	643
6.93.3.5	tightMarshal1 . . . . .	644
6.93.3.6	tightMarshal2 . . . . .	644
6.93.3.7	tightUnmarshal . . . . .	645
6.94	activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller Class Reference . . . . .	645
6.94.1	Detailed Description . . . . .	646
6.94.2	Constructor & Destructor Documentation . . . . .	646
6.94.2.1	ActiveMQTempTopicMarshaller . . . . .	646
6.94.2.2	~ActiveMQTempTopicMarshaller . . . . .	646
6.94.3	Member Function Documentation . . . . .	646
6.94.3.1	createObject . . . . .	646
6.94.3.2	getDataStructureType . . . . .	647



6.94.3.3	looseMarshal . . . . .	647
6.94.3.4	looseUnmarshal . . . . .	647
6.94.3.5	tightMarshal1 . . . . .	648
6.94.3.6	tightMarshal2 . . . . .	648
6.94.3.7	tightUnmarshal . . . . .	649
6.95	activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller	
	Class Reference . . . . .	649
6.95.1	Detailed Description . . . . .	650
6.95.2	Constructor & Destructor Documentation . . . . .	651
6.95.2.1	ActiveMQTempTopicMarshaller . . . . .	651
6.95.2.2	~ActiveMQTempTopicMarshaller . . . . .	651
6.95.3	Member Function Documentation . . . . .	651
6.95.3.1	createObject . . . . .	651
6.95.3.2	getDataStructureType . . . . .	651
6.95.3.3	looseMarshal . . . . .	651
6.95.3.4	looseUnmarshal . . . . .	652
6.95.3.5	tightMarshal1 . . . . .	652
6.95.3.6	tightMarshal2 . . . . .	653
6.95.3.7	tightUnmarshal . . . . .	653
6.96	activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	
	Class Reference . . . . .	654
6.96.1	Detailed Description . . . . .	655
6.96.2	Constructor & Destructor Documentation . . . . .	655
6.96.2.1	ActiveMQTempTopicMarshaller . . . . .	655
6.96.2.2	~ActiveMQTempTopicMarshaller . . . . .	655
6.96.3	Member Function Documentation . . . . .	655
6.96.3.1	createObject . . . . .	655
6.96.3.2	getDataStructureType . . . . .	655
6.96.3.3	looseMarshal . . . . .	655
6.96.3.4	looseUnmarshal . . . . .	656
6.96.3.5	tightMarshal1 . . . . .	656
6.96.3.6	tightMarshal2 . . . . .	657
6.96.3.7	tightUnmarshal . . . . .	657
6.97	activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller	
	Class Reference . . . . .	658
6.97.1	Detailed Description . . . . .	659
6.97.2	Constructor & Destructor Documentation . . . . .	659
6.97.2.1	ActiveMQTempTopicMarshaller . . . . .	659
6.97.2.2	~ActiveMQTempTopicMarshaller . . . . .	659
6.97.3	Member Function Documentation . . . . .	659
6.97.3.1	createObject . . . . .	659
6.97.3.2	getDataStructureType . . . . .	659
6.97.3.3	looseMarshal . . . . .	660
6.97.3.4	looseUnmarshal . . . . .	660
6.97.3.5	tightMarshal1 . . . . .	661
6.97.3.6	tightMarshal2 . . . . .	661
6.97.3.7	tightUnmarshal . . . . .	662
6.98	activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller	
	Class Reference . . . . .	662
6.98.1	Detailed Description . . . . .	663

6.98.2	Constructor & Destructor Documentation . . . . .	663
6.98.2.1	ActiveMQTempTopicMarshaller . . . . .	663
6.98.2.2	~ActiveMQTempTopicMarshaller . . . . .	663
6.98.3	Member Function Documentation . . . . .	663
6.98.3.1	createObject . . . . .	663
6.98.3.2	getDataStructureType . . . . .	664
6.98.3.3	looseMarshal . . . . .	664
6.98.3.4	looseUnmarshal . . . . .	664
6.98.3.5	tightMarshal1 . . . . .	665
6.98.3.6	tightMarshal2 . . . . .	665
6.98.3.7	tightUnmarshal . . . . .	666
6.99	activemq::commands::ActiveMQTextMessage Class Reference . . . . .	666
6.99.1	Constructor & Destructor Documentation . . . . .	668
6.99.1.1	ActiveMQTextMessage . . . . .	668
6.99.1.2	~ActiveMQTextMessage . . . . .	668
6.99.2	Member Function Documentation . . . . .	668
6.99.2.1	beforeMarshal . . . . .	668
6.99.2.2	clearBody . . . . .	668
6.99.2.3	clone . . . . .	668
6.99.2.4	cloneDataStructure . . . . .	669
6.99.2.5	copyDataStructure . . . . .	669
6.99.2.6	equals . . . . .	669
6.99.2.7	getDataStructureType . . . . .	669
6.99.2.8	getSize . . . . .	670
6.99.2.9	getText . . . . .	670
6.99.2.10	setText . . . . .	670
6.99.2.11	setText . . . . .	671
6.99.2.12	toString . . . . .	671
6.99.3	Field Documentation . . . . .	671
6.99.3.1	ID_ACTIVEMQTEXTMESSAGE . . . . .	671
6.99.3.2	text . . . . .	671
6.100	activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	
	Class Reference . . . . .	671
6.100.1	Detailed Description . . . . .	672
6.100.2	Constructor & Destructor Documentation . . . . .	673
6.100.2.1	ActiveMQTextMessageMarshaller . . . . .	673
6.100.2.2	~ActiveMQTextMessageMarshaller . . . . .	673
6.100.3	Member Function Documentation . . . . .	673
6.100.3.1	createObject . . . . .	673
6.100.3.2	getDataStructureType . . . . .	673
6.100.3.3	looseMarshal . . . . .	673
6.100.3.4	looseUnmarshal . . . . .	674
6.100.3.5	tightMarshal1 . . . . .	674
6.100.3.6	tightMarshal2 . . . . .	675
6.100.3.7	tightUnmarshal . . . . .	675
6.101	activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	
	Class Reference . . . . .	676
6.101.1	Detailed Description . . . . .	677
6.101.2	Constructor & Destructor Documentation . . . . .	677
6.101.2.1	ActiveMQTextMessageMarshaller . . . . .	677

6.101.2.2	~ActiveMQTextMessageMarshaller	677
6.101.3	Member Function Documentation	677
6.101.3.1	createObject	677
6.101.3.2	getDataStructureType	677
6.101.3.3	looseMarshal	677
6.101.3.4	looseUnmarshal	678
6.101.3.5	tightMarshal1	678
6.101.3.6	tightMarshal2	679
6.101.3.7	tightUnmarshal	679
6.102	activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller	
	Class Reference	680
6.102.1	Detailed Description	681
6.102.2	Constructor & Destructor Documentation	681
6.102.2.1	ActiveMQTextMessageMarshaller	681
6.102.2.2	~ActiveMQTextMessageMarshaller	681
6.102.3	Member Function Documentation	681
6.102.3.1	createObject	681
6.102.3.2	getDataStructureType	681
6.102.3.3	looseMarshal	682
6.102.3.4	looseUnmarshal	682
6.102.3.5	tightMarshal1	683
6.102.3.6	tightMarshal2	683
6.102.3.7	tightUnmarshal	684
6.103	activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	
	Class Reference	684
6.103.1	Detailed Description	685
6.103.2	Constructor & Destructor Documentation	685
6.103.2.1	ActiveMQTextMessageMarshaller	685
6.103.2.2	~ActiveMQTextMessageMarshaller	685
6.103.3	Member Function Documentation	685
6.103.3.1	createObject	685
6.103.3.2	getDataStructureType	686
6.103.3.3	looseMarshal	686
6.103.3.4	looseUnmarshal	686
6.103.3.5	tightMarshal1	687
6.103.3.6	tightMarshal2	687
6.103.3.7	tightUnmarshal	688
6.104	activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller	
	Class Reference	688
6.104.1	Detailed Description	689
6.104.2	Constructor & Destructor Documentation	690
6.104.2.1	ActiveMQTextMessageMarshaller	690
6.104.2.2	~ActiveMQTextMessageMarshaller	690
6.104.3	Member Function Documentation	690
6.104.3.1	createObject	690
6.104.3.2	getDataStructureType	690
6.104.3.3	looseMarshal	690
6.104.3.4	looseUnmarshal	691
6.104.3.5	tightMarshal1	691
6.104.3.6	tightMarshal2	692

6.104.3.7 tightUnmarshal . . . . .	692
6.105activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller	
Class Reference . . . . .	693
6.105.1 Detailed Description . . . . .	694
6.105.2 Constructor & Destructor Documentation . . . . .	694
6.105.2.1 ActiveMQTextMessageMarshaller . . . . .	694
6.105.2.2 ~ActiveMQTextMessageMarshaller . . . . .	694
6.105.3 Member Function Documentation . . . . .	694
6.105.3.1 createObject . . . . .	694
6.105.3.2 getDataStructureType . . . . .	694
6.105.3.3 looseMarshal . . . . .	694
6.105.3.4 looseUnmarshal . . . . .	695
6.105.3.5 tightMarshal1 . . . . .	695
6.105.3.6 tightMarshal2 . . . . .	696
6.105.3.7 tightUnmarshal . . . . .	696
6.106activemq::commands::ActiveMQTopic Class Reference . . . . .	697
6.106.1 Constructor & Destructor Documentation . . . . .	698
6.106.1.1 ActiveMQTopic . . . . .	698
6.106.1.2 ActiveMQTopic . . . . .	698
6.106.1.3 ~ActiveMQTopic . . . . .	698
6.106.2 Member Function Documentation . . . . .	698
6.106.2.1 clone . . . . .	698
6.106.2.2 cloneDataStructure . . . . .	698
6.106.2.3 copy . . . . .	699
6.106.2.4 copyDataStructure . . . . .	699
6.106.2.5 equals . . . . .	699
6.106.2.6 getCMSDestination . . . . .	699
6.106.2.7 getCMSProperties . . . . .	700
6.106.2.8 getDataStructureType . . . . .	700
6.106.2.9 getDestinationType . . . . .	700
6.106.2.10getTopicName . . . . .	700
6.106.2.11toString . . . . .	701
6.106.3 Field Documentation . . . . .	701
6.106.3.1 ID_ACTIVEMQTOPIC . . . . .	701
6.107activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	
Class Reference . . . . .	701
6.107.1 Detailed Description . . . . .	702
6.107.2 Constructor & Destructor Documentation . . . . .	702
6.107.2.1 ActiveMQTopicMarshaller . . . . .	702
6.107.2.2 ~ActiveMQTopicMarshaller . . . . .	702
6.107.3 Member Function Documentation . . . . .	702
6.107.3.1 createObject . . . . .	702
6.107.3.2 getDataStructureType . . . . .	703
6.107.3.3 looseMarshal . . . . .	703
6.107.3.4 looseUnmarshal . . . . .	703
6.107.3.5 tightMarshal1 . . . . .	704
6.107.3.6 tightMarshal2 . . . . .	704
6.107.3.7 tightUnmarshal . . . . .	705
6.108activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	
Class Reference . . . . .	705

6.108.1 Detailed Description . . . . .	706
6.108.2 Constructor & Destructor Documentation . . . . .	707
6.108.2.1 ActiveMQTopicMarshaller . . . . .	707
6.108.2.2 ~ActiveMQTopicMarshaller . . . . .	707
6.108.3 Member Function Documentation . . . . .	707
6.108.3.1 createObject . . . . .	707
6.108.3.2 getDataStructureType . . . . .	707
6.108.3.3 looseMarshal . . . . .	707
6.108.3.4 looseUnmarshal . . . . .	708
6.108.3.5 tightMarshal1 . . . . .	708
6.108.3.6 tightMarshal2 . . . . .	709
6.108.3.7 tightUnmarshal . . . . .	709
6.109activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller	
Class Reference . . . . .	710
6.109.1 Detailed Description . . . . .	711
6.109.2 Constructor & Destructor Documentation . . . . .	711
6.109.2.1 ActiveMQTopicMarshaller . . . . .	711
6.109.2.2 ~ActiveMQTopicMarshaller . . . . .	711
6.109.3 Member Function Documentation . . . . .	711
6.109.3.1 createObject . . . . .	711
6.109.3.2 getDataStructureType . . . . .	711
6.109.3.3 looseMarshal . . . . .	711
6.109.3.4 looseUnmarshal . . . . .	712
6.109.3.5 tightMarshal1 . . . . .	712
6.109.3.6 tightMarshal2 . . . . .	713
6.109.3.7 tightUnmarshal . . . . .	713
6.110activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller	
Class Reference . . . . .	714
6.110.1 Detailed Description . . . . .	715
6.110.2 Constructor & Destructor Documentation . . . . .	715
6.110.2.1 ActiveMQTopicMarshaller . . . . .	715
6.110.2.2 ~ActiveMQTopicMarshaller . . . . .	715
6.110.3 Member Function Documentation . . . . .	715
6.110.3.1 createObject . . . . .	715
6.110.3.2 getDataStructureType . . . . .	715
6.110.3.3 looseMarshal . . . . .	716
6.110.3.4 looseUnmarshal . . . . .	716
6.110.3.5 tightMarshal1 . . . . .	717
6.110.3.6 tightMarshal2 . . . . .	717
6.110.3.7 tightUnmarshal . . . . .	718
6.111activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller	
Class Reference . . . . .	718
6.111.1 Detailed Description . . . . .	719
6.111.2 Constructor & Destructor Documentation . . . . .	719
6.111.2.1 ActiveMQTopicMarshaller . . . . .	719
6.111.2.2 ~ActiveMQTopicMarshaller . . . . .	719
6.111.3 Member Function Documentation . . . . .	719
6.111.3.1 createObject . . . . .	719
6.111.3.2 getDataStructureType . . . . .	720
6.111.3.3 looseMarshal . . . . .	720

6.111.3.4 looseUnmarshal . . . . .	720
6.111.3.5 tightMarshal1 . . . . .	721
6.111.3.6 tightMarshal2 . . . . .	721
6.111.3.7 tightUnmarshal . . . . .	722
6.112activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller	
Class Reference . . . . .	722
6.112.1 Detailed Description . . . . .	723
6.112.2 Constructor & Destructor Documentation . . . . .	724
6.112.2.1 ActiveMQTopicMarshaller . . . . .	724
6.112.2.2 ~ActiveMQTopicMarshaller . . . . .	724
6.112.3 Member Function Documentation . . . . .	724
6.112.3.1 createObject . . . . .	724
6.112.3.2 getDataStructureType . . . . .	724
6.112.3.3 looseMarshal . . . . .	724
6.112.3.4 looseUnmarshal . . . . .	725
6.112.3.5 tightMarshal1 . . . . .	725
6.112.3.6 tightMarshal2 . . . . .	726
6.112.3.7 tightUnmarshal . . . . .	726
6.113activemq::core::ActiveMQTransactionContext Class Reference . . . .	727
6.113.1 Detailed Description . . . . .	728
6.113.2 Constructor & Destructor Documentation . . . . .	728
6.113.2.1 ActiveMQTransactionContext . . . . .	728
6.113.2.2 ~ActiveMQTransactionContext . . . . .	728
6.113.3 Member Function Documentation . . . . .	728
6.113.3.1 addSynchronization . . . . .	728
6.113.3.2 begin . . . . .	728
6.113.3.3 commit . . . . .	729
6.113.3.4 getTransactionId . . . . .	729
6.113.3.5 isInTransaction . . . . .	729
6.113.3.6 removeSynchronization . . . . .	729
6.113.3.7 rollback . . . . .	730
6.114decaf::util::zip::Adler32 Class Reference . . . . .	730
6.114.1 Detailed Description . . . . .	731
6.114.2 Constructor & Destructor Documentation . . . . .	731
6.114.2.1 Adler32 . . . . .	731
6.114.2.2 ~Adler32 . . . . .	731
6.114.3 Member Function Documentation . . . . .	731
6.114.3.1 getValue . . . . .	731
6.114.3.2 reset . . . . .	731
6.114.3.3 update . . . . .	731
6.114.3.4 update . . . . .	732
6.114.3.5 update . . . . .	732
6.114.3.6 update . . . . .	732
6.115decaf::lang::Appendable Class Reference . . . . .	733
6.115.1 Detailed Description . . . . .	733
6.115.2 Constructor & Destructor Documentation . . . . .	734
6.115.2.1 ~Appendable . . . . .	734
6.115.3 Member Function Documentation . . . . .	734
6.115.3.1 append . . . . .	734
6.115.3.2 append . . . . .	734

6.115.3.3 append . . . . .	735
6.116decaf::internal::AprPool Class Reference . . . . .	735
6.116.1 Detailed Description . . . . .	736
6.116.2 Constructor & Destructor Documentation . . . . .	736
6.116.2.1 AprPool . . . . .	736
6.116.2.2 ~AprPool . . . . .	736
6.116.3 Member Function Documentation . . . . .	736
6.116.3.1 cleanup . . . . .	736
6.116.3.2 getAprPool . . . . .	736
6.116.3.3 getGlobalPool . . . . .	736
6.117decaf::lang::ArrayPointer< T, REFCOUNTER > Class Template Reference . . . . .	736
6.117.1 Detailed Description . . . . .	738
6.117.2 Member Typedef Documentation . . . . .	739
6.117.2.1 ConstReferenceType . . . . .	739
6.117.2.2 CounterType . . . . .	739
6.117.2.3 PointerType . . . . .	739
6.117.2.4 ReferenceType . . . . .	739
6.117.3 Constructor & Destructor Documentation . . . . .	739
6.117.3.1 ArrayPointer . . . . .	739
6.117.3.2 ArrayPointer . . . . .	739
6.117.3.3 ArrayPointer . . . . .	740
6.117.3.4 ArrayPointer . . . . .	740
6.117.3.5 ~ArrayPointer . . . . .	740
6.117.4 Member Function Documentation . . . . .	740
6.117.4.1 clone . . . . .	740
6.117.4.2 get . . . . .	741
6.117.4.3 length . . . . .	741
6.117.4.4 operator! . . . . .	742
6.117.4.5 operator!= . . . . .	742
6.117.4.6 operator= . . . . .	742
6.117.4.7 operator= . . . . .	742
6.117.4.8 operator== . . . . .	742
6.117.4.9 operator[] . . . . .	742
6.117.4.10operator[] . . . . .	743
6.117.4.11release . . . . .	743
6.117.4.12reset . . . . .	743
6.117.4.13swap . . . . .	743
6.117.5 Friends And Related Function Documentation . . . . .	744
6.117.5.1 operator!= . . . . .	744
6.117.5.2 operator!= . . . . .	744
6.117.5.3 operator== . . . . .	744
6.117.5.4 operator== . . . . .	744
6.118decaf::lang::ArrayPointerComparator< T, R > Class Template Reference	744
6.118.1 Detailed Description . . . . .	745
6.118.2 Member Function Documentation . . . . .	745
6.118.2.1 compare . . . . .	745
6.118.2.2 operator() . . . . .	745
6.119decaf::util::concurrent::atomic::AtomicBoolean Class Reference . . .	745
6.119.1 Detailed Description . . . . .	746

6.119.2 Constructor & Destructor Documentation . . . . .	746
6.119.2.1 AtomicBoolean . . . . .	746
6.119.2.2 AtomicBoolean . . . . .	746
6.119.2.3 ~AtomicBoolean . . . . .	747
6.119.3 Member Function Documentation . . . . .	747
6.119.3.1 compareAndSet . . . . .	747
6.119.3.2 get . . . . .	747
6.119.3.3 getAndSet . . . . .	747
6.119.3.4 set . . . . .	747
6.119.3.5 toString . . . . .	748
6.120decaf::util::concurrent::atomic::AtomicInteger Class Reference . . . . .	748
6.120.1 Detailed Description . . . . .	749
6.120.2 Constructor & Destructor Documentation . . . . .	750
6.120.2.1 AtomicInteger . . . . .	750
6.120.2.2 AtomicInteger . . . . .	750
6.120.2.3 ~AtomicInteger . . . . .	750
6.120.3 Member Function Documentation . . . . .	750
6.120.3.1 addAndGet . . . . .	750
6.120.3.2 compareAndSet . . . . .	750
6.120.3.3 decrementAndGet . . . . .	751
6.120.3.4 doubleValue . . . . .	751
6.120.3.5 floatValue . . . . .	751
6.120.3.6 get . . . . .	751
6.120.3.7 getAndAdd . . . . .	751
6.120.3.8 getAndDecrement . . . . .	752
6.120.3.9 getAndIncrement . . . . .	752
6.120.3.10getAndSet . . . . .	752
6.120.3.11incrementAndGet . . . . .	752
6.120.3.12intValue . . . . .	753
6.120.3.13longValue . . . . .	753
6.120.3.14set . . . . .	753
6.120.3.15toString . . . . .	753
6.121decaf::util::concurrent::atomic::AtomicRefCounter Class Reference . . . . .	754
6.121.1 Constructor & Destructor Documentation . . . . .	755
6.121.1.1 AtomicRefCounter . . . . .	755
6.121.1.2 AtomicRefCounter . . . . .	755
6.121.1.3 ~AtomicRefCounter . . . . .	755
6.121.2 Member Function Documentation . . . . .	755
6.121.2.1 release . . . . .	755
6.121.2.2 swap . . . . .	756
6.122decaf::util::concurrent::atomic::AtomicReference< T > Class Template	
Reference . . . . .	756
6.122.1 Detailed Description . . . . .	757
6.122.2 Constructor & Destructor Documentation . . . . .	757
6.122.2.1 AtomicReference . . . . .	757
6.122.2.2 AtomicReference . . . . .	757
6.122.2.3 ~AtomicReference . . . . .	757
6.122.3 Member Function Documentation . . . . .	757
6.122.3.1 compareAndSet . . . . .	757
6.122.3.2 get . . . . .	758



6.122.3.3	getAndSet . . . . .	758
6.122.3.4	set . . . . .	758
6.122.3.5	toString . . . . .	758
6.123	activemq::transport::failover::BackupTransport Class Reference . . .	759
6.123.1	Constructor & Destructor Documentation . . . . .	760
6.123.1.1	BackupTransport . . . . .	760
6.123.1.2	~BackupTransport . . . . .	760
6.123.2	Member Function Documentation . . . . .	760
6.123.2.1	getTransport . . . . .	760
6.123.2.2	getUri . . . . .	760
6.123.2.3	isClosed . . . . .	760
6.123.2.4	onException . . . . .	760
6.123.2.5	setClosed . . . . .	761
6.123.2.6	setTransport . . . . .	761
6.123.2.7	setUri . . . . .	761
6.124	activemq::transport::failover::BackupTransportPool Class Reference .	761
6.124.1	Constructor & Destructor Documentation . . . . .	763
6.124.1.1	BackupTransportPool . . . . .	763
6.124.1.2	BackupTransportPool . . . . .	763
6.124.1.3	~BackupTransportPool . . . . .	763
6.124.2	Member Function Documentation . . . . .	763
6.124.2.1	getBackup . . . . .	763
6.124.2.2	getBackupPoolSize . . . . .	763
6.124.2.3	isEnabled . . . . .	763
6.124.2.4	isPending . . . . .	764
6.124.2.5	iterate . . . . .	764
6.124.2.6	setBackupPoolSize . . . . .	764
6.124.2.7	setEnabled . . . . .	764
6.124.3	Friends And Related Function Documentation . . . . .	764
6.124.3.1	BackupTransport . . . . .	764
6.125	activemq::commands::BaseCommand Class Reference . . . . .	764
6.125.1	Constructor & Destructor Documentation . . . . .	766
6.125.1.1	BaseCommand . . . . .	766
6.125.1.2	~BaseCommand . . . . .	766
6.125.2	Member Function Documentation . . . . .	766
6.125.2.1	copyDataStructure . . . . .	766
6.125.2.2	equals . . . . .	766
6.125.2.3	getCommandId . . . . .	767
6.125.2.4	isBrokerInfo . . . . .	767
6.125.2.5	isConnectionInfo . . . . .	768
6.125.2.6	isConsumerInfo . . . . .	768
6.125.2.7	isKeepAliveInfo . . . . .	768
6.125.2.8	isMessage . . . . .	768
6.125.2.9	isMessageAck . . . . .	768
6.125.2.10	isMessageDispatch . . . . .	768
6.125.2.11	isMessageDispatchNotification . . . . .	768
6.125.2.12	isProducerAck . . . . .	769
6.125.2.13	isProducerInfo . . . . .	769
6.125.2.14	isRemoveInfo . . . . .	769
6.125.2.15	isRemoveSubscriptionInfo . . . . .	769

6.125.2.16	gsResponse . . . . .	769
6.125.2.17	gsResponseRequired . . . . .	769
6.125.2.18	gsShutdownInfo . . . . .	770
6.125.2.19	gsTransactionInfo . . . . .	770
6.125.2.20	gsWireFormatInfo . . . . .	770
6.125.2.21	setCommandId . . . . .	770
6.125.2.22	setResponseRequired . . . . .	770
6.125.2.23	toString . . . . .	770
6.126	activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller	
	Class Reference . . . . .	771
6.126.1	Detailed Description . . . . .	772
6.126.2	Constructor & Destructor Documentation . . . . .	772
6.126.2.1	BaseCommandMarshaller . . . . .	772
6.126.2.2	~BaseCommandMarshaller . . . . .	772
6.126.3	Member Function Documentation . . . . .	772
6.126.3.1	looseMarshal . . . . .	772
6.126.3.2	looseUnmarshal . . . . .	774
6.126.3.3	tightMarshal1 . . . . .	775
6.126.3.4	tightMarshal2 . . . . .	776
6.126.3.5	tightUnmarshal . . . . .	777
6.127	activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller	
	Class Reference . . . . .	778
6.127.1	Detailed Description . . . . .	779
6.127.2	Constructor & Destructor Documentation . . . . .	780
6.127.2.1	BaseCommandMarshaller . . . . .	780
6.127.2.2	~BaseCommandMarshaller . . . . .	780
6.127.3	Member Function Documentation . . . . .	780
6.127.3.1	looseMarshal . . . . .	780
6.127.3.2	looseUnmarshal . . . . .	781
6.127.3.3	tightMarshal1 . . . . .	782
6.127.3.4	tightMarshal2 . . . . .	783
6.127.3.5	tightUnmarshal . . . . .	784
6.128	activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller	
	Class Reference . . . . .	786
6.128.1	Detailed Description . . . . .	787
6.128.2	Constructor & Destructor Documentation . . . . .	787
6.128.2.1	BaseCommandMarshaller . . . . .	787
6.128.2.2	~BaseCommandMarshaller . . . . .	787
6.128.3	Member Function Documentation . . . . .	787
6.128.3.1	looseMarshal . . . . .	787
6.128.3.2	looseUnmarshal . . . . .	788
6.128.3.3	tightMarshal1 . . . . .	789
6.128.3.4	tightMarshal2 . . . . .	790
6.128.3.5	tightUnmarshal . . . . .	792
6.129	activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller	
	Class Reference . . . . .	793
6.129.1	Detailed Description . . . . .	794
6.129.2	Constructor & Destructor Documentation . . . . .	794
6.129.2.1	BaseCommandMarshaller . . . . .	794
6.129.2.2	~BaseCommandMarshaller . . . . .	794

6.129.3 Member Function Documentation . . . . .	794
6.129.3.1 looseMarshal . . . . .	794
6.129.3.2 looseUnmarshal . . . . .	795
6.129.3.3 tightMarshal1 . . . . .	796
6.129.3.4 tightMarshal2 . . . . .	797
6.129.3.5 tightUnmarshal . . . . .	799
6.130activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller	
Class Reference . . . . .	800
6.130.1 Detailed Description . . . . .	801
6.130.2 Constructor & Destructor Documentation . . . . .	801
6.130.2.1 BaseCommandMarshaller . . . . .	801
6.130.2.2 ~BaseCommandMarshaller . . . . .	801
6.130.3 Member Function Documentation . . . . .	801
6.130.3.1 looseMarshal . . . . .	801
6.130.3.2 looseUnmarshal . . . . .	802
6.130.3.3 tightMarshal1 . . . . .	803
6.130.3.4 tightMarshal2 . . . . .	804
6.130.3.5 tightUnmarshal . . . . .	806
6.131activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller	
Class Reference . . . . .	807
6.131.1 Detailed Description . . . . .	808
6.131.2 Constructor & Destructor Documentation . . . . .	808
6.131.2.1 BaseCommandMarshaller . . . . .	808
6.131.2.2 ~BaseCommandMarshaller . . . . .	808
6.131.3 Member Function Documentation . . . . .	808
6.131.3.1 looseMarshal . . . . .	808
6.131.3.2 looseUnmarshal . . . . .	809
6.131.3.3 tightMarshal1 . . . . .	810
6.131.3.4 tightMarshal2 . . . . .	811
6.131.3.5 tightUnmarshal . . . . .	813
6.132activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller	
Class Reference . . . . .	814
6.132.1 Detailed Description . . . . .	820
6.132.2 Constructor & Destructor Documentation . . . . .	820
6.132.2.1 ~BaseDataStreamMarshaller . . . . .	820
6.132.3 Member Function Documentation . . . . .	820
6.132.3.1 looseMarshal . . . . .	820
6.132.3.2 looseMarshalBrokerError . . . . .	821
6.132.3.3 looseMarshalCachedObject . . . . .	821
6.132.3.4 looseMarshalLong . . . . .	821
6.132.3.5 looseMarshalNestedObject . . . . .	822
6.132.3.6 looseMarshalObjectArray . . . . .	822
6.132.3.7 looseMarshalString . . . . .	823
6.132.3.8 looseUnmarshal . . . . .	823
6.132.3.9 looseUnmarshalBrokerError . . . . .	823
6.132.3.10looseUnmarshalByteArray . . . . .	824
6.132.3.11looseUnmarshalCachedObject . . . . .	824
6.132.3.12looseUnmarshalConstByteArray . . . . .	825
6.132.3.13looseUnmarshalLong . . . . .	825
6.132.3.14looseUnmarshalNestedObject . . . . .	825

6.132.3.15	looseUnmarshalString . . . . .	826
6.132.3.16	readAsciiString . . . . .	826
6.132.3.17	tightMarshal1 . . . . .	827
6.132.3.18	tightMarshal2 . . . . .	827
6.132.3.19	tightMarshalBrokerError1 . . . . .	827
6.132.3.20	tightMarshalBrokerError2 . . . . .	828
6.132.3.21	tightMarshalCachedObject1 . . . . .	828
6.132.3.22	tightMarshalCachedObject2 . . . . .	829
6.132.3.23	tightMarshalLong1 . . . . .	829
6.132.3.24	tightMarshalLong2 . . . . .	829
6.132.3.25	tightMarshalNestedObject1 . . . . .	830
6.132.3.26	tightMarshalNestedObject2 . . . . .	830
6.132.3.27	tightMarshalObjectArray1 . . . . .	831
6.132.3.28	tightMarshalObjectArray2 . . . . .	831
6.132.3.29	tightMarshalString1 . . . . .	832
6.132.3.30	tightMarshalString2 . . . . .	832
6.132.3.31	tightUnmarshal . . . . .	832
6.132.3.32	tightUnmarshalBrokerError . . . . .	833
6.132.3.33	tightUnmarshalByteArray . . . . .	833
6.132.3.34	tightUnmarshalCachedObject . . . . .	834
6.132.3.35	tightUnmarshalConstByteArray . . . . .	834
6.132.3.36	tightUnmarshalLong . . . . .	835
6.132.3.37	tightUnmarshalNestedObject . . . . .	835
6.132.3.38	tightUnmarshalString . . . . .	835
6.132.3.39	toHexFromBytes . . . . .	836
6.132.3.40	toString . . . . .	836
6.132.3.41	toString . . . . .	836
6.132.3.42	toString . . . . .	837
6.133	activemq::commands::BaseDataStructure Class Reference . . . . .	837
6.133.1	Constructor & Destructor Documentation . . . . .	838
6.133.1.1	~BaseDataStructure . . . . .	838
6.133.2	Member Function Documentation . . . . .	838
6.133.2.1	afterMarshal . . . . .	838
6.133.2.2	afterUnmarshal . . . . .	839
6.133.2.3	beforeMarshal . . . . .	839
6.133.2.4	beforeUnmarshal . . . . .	839
6.133.2.5	copyDataStructure . . . . .	839
6.133.2.6	equals . . . . .	840
6.133.2.7	getMarshaledForm . . . . .	840
6.133.2.8	isMarshalAware . . . . .	840
6.133.2.9	setMarshaledForm . . . . .	841
6.133.2.10	toString . . . . .	841
6.134	binary_function Class Reference . . . . .	842
6.135	decaf::net::BindException Class Reference . . . . .	842
6.135.1	Constructor & Destructor Documentation . . . . .	843
6.135.1.1	BindException . . . . .	843
6.135.1.2	BindException . . . . .	843
6.135.1.3	BindException . . . . .	843
6.135.1.4	BindException . . . . .	843
6.135.1.5	BindException . . . . .	844

6.135.1.6	BindException . . . . .	844
6.135.1.7	~BindException . . . . .	844
6.135.2	Member Function Documentation . . . . .	844
6.135.2.1	clone . . . . .	844
6.136	decaf::io::BlockingByteArrayInputStream Class Reference . . . . .	845
6.136.1	Detailed Description . . . . .	846
6.136.2	Constructor & Destructor Documentation . . . . .	846
6.136.2.1	BlockingByteArrayInputStream . . . . .	846
6.136.2.2	BlockingByteArrayInputStream . . . . .	846
6.136.2.3	~BlockingByteArrayInputStream . . . . .	847
6.136.3	Member Function Documentation . . . . .	847
6.136.3.1	available . . . . .	847
6.136.3.2	close . . . . .	847
6.136.3.3	doReadArrayBounded . . . . .	847
6.136.3.4	doReadByte . . . . .	847
6.136.3.5	setByteArray . . . . .	848
6.136.3.6	skip . . . . .	848
6.137	decaf::util::concurrent::BlockingQueue< E > Class Template Reference . . . . .	848
6.137.1	Detailed Description . . . . .	849
6.137.2	Constructor & Destructor Documentation . . . . .	852
6.137.2.1	~BlockingQueue . . . . .	852
6.137.3	Member Function Documentation . . . . .	852
6.137.3.1	drainTo . . . . .	852
6.137.3.2	drainTo . . . . .	852
6.137.3.3	offer . . . . .	853
6.137.3.4	poll . . . . .	854
6.137.3.5	put . . . . .	854
6.137.3.6	remainingCapacity . . . . .	855
6.137.3.7	take . . . . .	855
6.138	decaf::lang::Boolean Class Reference . . . . .	856
6.138.1	Constructor & Destructor Documentation . . . . .	857
6.138.1.1	Boolean . . . . .	857
6.138.1.2	Boolean . . . . .	857
6.138.1.3	~Boolean . . . . .	857
6.138.2	Member Function Documentation . . . . .	857
6.138.2.1	booleanValue . . . . .	857
6.138.2.2	compareTo . . . . .	857
6.138.2.3	compareTo . . . . .	858
6.138.2.4	equals . . . . .	858
6.138.2.5	equals . . . . .	858
6.138.2.6	operator< . . . . .	858
6.138.2.7	operator< . . . . .	859
6.138.2.8	operator== . . . . .	859
6.138.2.9	operator== . . . . .	859
6.138.2.10	parseBoolean . . . . .	860
6.138.2.11	toString . . . . .	860
6.138.2.12	toString . . . . .	860
6.138.2.13	valueOf . . . . .	860
6.138.2.14	valueOf . . . . .	861
6.138.3	Field Documentation . . . . .	861

6.138.3.1	_FALSE	861
6.138.3.2	_TRUE	861
6.139	activemq::commands::BooleanExpression Class Reference	861
6.139.1	Constructor & Destructor Documentation	862
6.139.1.1	BooleanExpression	862
6.139.1.2	~BooleanExpression	862
6.139.2	Member Function Documentation	862
6.139.2.1	cloneDataStructure	862
6.139.2.2	copyDataStructure	862
6.139.2.3	equals	862
6.139.2.4	toString	863
6.140	activemq::wireformat::openwire::utils::BooleanStream Class Reference	863
6.140.1	Detailed Description	864
6.140.2	Constructor & Destructor Documentation	864
6.140.2.1	BooleanStream	864
6.140.2.2	~BooleanStream	864
6.140.3	Member Function Documentation	864
6.140.3.1	clear	864
6.140.3.2	marshal	864
6.140.3.3	marshal	865
6.140.3.4	marshalledSize	865
6.140.3.5	readBoolean	865
6.140.3.6	unmarshal	865
6.140.3.7	writeBoolean	865
6.141	decaf::util::concurrent::BrokenBarrierException Class Reference	866
6.141.1	Constructor & Destructor Documentation	867
6.141.1.1	BrokenBarrierException	867
6.141.1.2	BrokenBarrierException	867
6.141.1.3	BrokenBarrierException	867
6.141.1.4	BrokenBarrierException	867
6.141.1.5	BrokenBarrierException	867
6.141.1.6	BrokenBarrierException	868
6.141.1.7	~BrokenBarrierException	868
6.141.2	Member Function Documentation	868
6.141.2.1	clone	868
6.142	activemq::commands::BrokerError Class Reference	868
6.142.1	Detailed Description	870
6.142.2	Constructor & Destructor Documentation	870
6.142.2.1	BrokerError	870
6.142.2.2	~BrokerError	870
6.142.3	Member Function Documentation	870
6.142.3.1	cloneDataStructure	870
6.142.3.2	copyDataStructure	870
6.142.3.3	getCause	871
6.142.3.4	getDataStructureType	871
6.142.3.5	getExceptionClass	871
6.142.3.6	getMessage	871
6.142.3.7	getStackTraceElements	871
6.142.3.8	setCause	872
6.142.3.9	setExceptionClass	872

6.142.3.10	setMessage . . . . .	872
6.142.3.11	setStackTraceElements . . . . .	872
6.142.3.12	visit . . . . .	873
6.143	activemq::exceptions::BrokerException Class Reference . . . . .	873
6.143.1	Constructor & Destructor Documentation . . . . .	874
6.143.1.1	BrokerException . . . . .	874
6.143.1.2	BrokerException . . . . .	874
6.143.1.3	BrokerException . . . . .	874
6.143.1.4	BrokerException . . . . .	874
6.143.1.5	BrokerException . . . . .	874
6.143.1.6	~BrokerException . . . . .	874
6.143.2	Member Function Documentation . . . . .	874
6.143.2.1	clone . . . . .	874
6.144	activemq::commands::BrokerId Class Reference . . . . .	874
6.144.1	Member Typedef Documentation . . . . .	876
6.144.1.1	COMPARATOR . . . . .	876
6.144.2	Constructor & Destructor Documentation . . . . .	876
6.144.2.1	BrokerId . . . . .	876
6.144.2.2	BrokerId . . . . .	876
6.144.2.3	~BrokerId . . . . .	876
6.144.3	Member Function Documentation . . . . .	876
6.144.3.1	cloneDataStructure . . . . .	876
6.144.3.2	compareTo . . . . .	876
6.144.3.3	copyDataStructure . . . . .	876
6.144.3.4	equals . . . . .	876
6.144.3.5	equals . . . . .	877
6.144.3.6	getDataStructureType . . . . .	877
6.144.3.7	getValue . . . . .	877
6.144.3.8	getValue . . . . .	877
6.144.3.9	operator< . . . . .	877
6.144.3.10	operator= . . . . .	877
6.144.3.11	operator== . . . . .	877
6.144.3.12	setValue . . . . .	877
6.144.3.13	toString . . . . .	877
6.144.4	Field Documentation . . . . .	878
6.144.4.1	ID_BROKERID . . . . .	878
6.144.4.2	value . . . . .	878
6.145	activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller Class Reference . . . . .	878
6.145.1	Detailed Description . . . . .	879
6.145.2	Constructor & Destructor Documentation . . . . .	879
6.145.2.1	BrokerIdMarshaller . . . . .	879
6.145.2.2	~BrokerIdMarshaller . . . . .	879
6.145.3	Member Function Documentation . . . . .	879
6.145.3.1	createObject . . . . .	879
6.145.3.2	getDataStructureType . . . . .	879
6.145.3.3	looseMarshal . . . . .	880
6.145.3.4	looseUnmarshal . . . . .	880
6.145.3.5	tightMarshal1 . . . . .	880
6.145.3.6	tightMarshal2 . . . . .	881

6.145.3.7 tightUnmarshal . . . . .	881
6.146activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller Class	
Reference . . . . .	882
6.146.1 Detailed Description . . . . .	883
6.146.2 Constructor & Destructor Documentation . . . . .	883
6.146.2.1 BrokerIdMarshaller . . . . .	883
6.146.2.2 ~BrokerIdMarshaller . . . . .	883
6.146.3 Member Function Documentation . . . . .	883
6.146.3.1 createObject . . . . .	883
6.146.3.2 getDataStructureType . . . . .	883
6.146.3.3 looseMarshal . . . . .	884
6.146.3.4 looseUnmarshal . . . . .	884
6.146.3.5 tightMarshal1 . . . . .	885
6.146.3.6 tightMarshal2 . . . . .	885
6.146.3.7 tightUnmarshal . . . . .	886
6.147activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller Class	
Reference . . . . .	886
6.147.1 Detailed Description . . . . .	887
6.147.2 Constructor & Destructor Documentation . . . . .	887
6.147.2.1 BrokerIdMarshaller . . . . .	887
6.147.2.2 ~BrokerIdMarshaller . . . . .	887
6.147.3 Member Function Documentation . . . . .	887
6.147.3.1 createObject . . . . .	887
6.147.3.2 getDataStructureType . . . . .	888
6.147.3.3 looseMarshal . . . . .	888
6.147.3.4 looseUnmarshal . . . . .	888
6.147.3.5 tightMarshal1 . . . . .	889
6.147.3.6 tightMarshal2 . . . . .	889
6.147.3.7 tightUnmarshal . . . . .	890
6.148activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller Class	
Reference . . . . .	890
6.148.1 Detailed Description . . . . .	891
6.148.2 Constructor & Destructor Documentation . . . . .	891
6.148.2.1 BrokerIdMarshaller . . . . .	891
6.148.2.2 ~BrokerIdMarshaller . . . . .	891
6.148.3 Member Function Documentation . . . . .	891
6.148.3.1 createObject . . . . .	891
6.148.3.2 getDataStructureType . . . . .	892
6.148.3.3 looseMarshal . . . . .	892
6.148.3.4 looseUnmarshal . . . . .	892
6.148.3.5 tightMarshal1 . . . . .	893
6.148.3.6 tightMarshal2 . . . . .	893
6.148.3.7 tightUnmarshal . . . . .	894
6.149activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller Class	
Reference . . . . .	894
6.149.1 Detailed Description . . . . .	895
6.149.2 Constructor & Destructor Documentation . . . . .	895
6.149.2.1 BrokerIdMarshaller . . . . .	895
6.149.2.2 ~BrokerIdMarshaller . . . . .	895
6.149.3 Member Function Documentation . . . . .	895



6.149.3.1	createObject	895
6.149.3.2	getDataStructureType	896
6.149.3.3	looseMarshal	896
6.149.3.4	looseUnmarshal	896
6.149.3.5	tightMarshal1	897
6.149.3.6	tightMarshal2	897
6.149.3.7	tightUnmarshal	898
6.150	activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller Class	
	Reference	898
6.150.1	Detailed Description	899
6.150.2	Constructor & Destructor Documentation	899
6.150.2.1	BrokerIdMarshaller	899
6.150.2.2	~BrokerIdMarshaller	899
6.150.3	Member Function Documentation	899
6.150.3.1	createObject	899
6.150.3.2	getDataStructureType	900
6.150.3.3	looseMarshal	900
6.150.3.4	looseUnmarshal	900
6.150.3.5	tightMarshal1	901
6.150.3.6	tightMarshal2	901
6.150.3.7	tightUnmarshal	902
6.151	activemq::commands::BrokerInfo Class Reference	902
6.151.1	Constructor & Destructor Documentation	904
6.151.1.1	BrokerInfo	904
6.151.1.2	~BrokerInfo	904
6.151.2	Member Function Documentation	904
6.151.2.1	cloneDataStructure	904
6.151.2.2	copyDataStructure	904
6.151.2.3	equals	905
6.151.2.4	getBrokerId	906
6.151.2.5	getBrokerId	906
6.151.2.6	getBrokerName	906
6.151.2.7	getBrokerName	906
6.151.2.8	getBrokerUploadUrl	906
6.151.2.9	getBrokerUploadUrl	906
6.151.2.10	getBrokerURL	906
6.151.2.11	getBrokerURL	906
6.151.2.12	getConnectionId	906
6.151.2.13	getDataStructureType	906
6.151.2.14	getNetworkProperties	907
6.151.2.15	getNetworkProperties	907
6.151.2.16	getPeerBrokerInfos	907
6.151.2.17	getPeerBrokerInfos	907
6.151.2.18	isBrokerInfo	907
6.151.2.19	isDuplexConnection	908
6.151.2.20	isFaultTolerantConfiguration	908
6.151.2.21	isMasterBroker	908
6.151.2.22	isNetworkConnection	908
6.151.2.23	isSlaveBroker	908
6.151.2.24	setBrokerId	908

6.151.2.25	setBrokerName	908
6.151.2.26	setBrokerUploadUrl	908
6.151.2.27	setBrokerURL	908
6.151.2.28	setConnectionId	908
6.151.2.29	setDuplexConnection	908
6.151.2.30	setFaultTolerantConfiguration	908
6.151.2.31	setMasterBroker	908
6.151.2.32	setNetworkConnection	908
6.151.2.33	setNetworkProperties	908
6.151.2.34	setPeerBrokerInfos	908
6.151.2.35	setSlaveBroker	908
6.151.2.36	toString	908
6.151.2.37	visit	909
6.151.3	Field Documentation	910
6.151.3.1	brokerId	910
6.151.3.2	brokerName	910
6.151.3.3	brokerUploadUrl	910
6.151.3.4	brokerURL	910
6.151.3.5	connectionId	910
6.151.3.6	duplexConnection	910
6.151.3.7	faultTolerantConfiguration	910
6.151.3.8	ID_BROKERINFO	910
6.151.3.9	masterBroker	910
6.151.3.10	networkConnection	910
6.151.3.11	networkProperties	910
6.151.3.12	peerBrokerInfos	910
6.151.3.13	slaveBroker	910
6.152	activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller Class	
	Reference	910
6.152.1	Detailed Description	911
6.152.2	Constructor & Destructor Documentation	912
6.152.2.1	BrokerInfoMarshaller	912
6.152.2.2	~BrokerInfoMarshaller	912
6.152.3	Member Function Documentation	912
6.152.3.1	createObject	912
6.152.3.2	getDataStructureType	912
6.152.3.3	looseMarshal	912
6.152.3.4	looseUnmarshal	913
6.152.3.5	tightMarshal1	913
6.152.3.6	tightMarshal2	914
6.152.3.7	tightUnmarshal	914
6.153	activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller Class	
	Reference	915
6.153.1	Detailed Description	916
6.153.2	Constructor & Destructor Documentation	916
6.153.2.1	BrokerInfoMarshaller	916
6.153.2.2	~BrokerInfoMarshaller	916
6.153.3	Member Function Documentation	916
6.153.3.1	createObject	916
6.153.3.2	getDataStructureType	916

6.153.3.3	looseMarshal . . . . .	916
6.153.3.4	looseUnmarshal . . . . .	917
6.153.3.5	tightMarshal1 . . . . .	917
6.153.3.6	tightMarshal2 . . . . .	918
6.153.3.7	tightUnmarshal . . . . .	918
6.154	activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller Class	
	Reference . . . . .	919
6.154.1	Detailed Description . . . . .	920
6.154.2	Constructor & Destructor Documentation . . . . .	920
6.154.2.1	BrokerInfoMarshaller . . . . .	920
6.154.2.2	~BrokerInfoMarshaller . . . . .	920
6.154.3	Member Function Documentation . . . . .	920
6.154.3.1	createObject . . . . .	920
6.154.3.2	getDataStructureType . . . . .	920
6.154.3.3	looseMarshal . . . . .	921
6.154.3.4	looseUnmarshal . . . . .	921
6.154.3.5	tightMarshal1 . . . . .	922
6.154.3.6	tightMarshal2 . . . . .	922
6.154.3.7	tightUnmarshal . . . . .	923
6.155	activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller Class	
	Reference . . . . .	923
6.155.1	Detailed Description . . . . .	924
6.155.2	Constructor & Destructor Documentation . . . . .	924
6.155.2.1	BrokerInfoMarshaller . . . . .	924
6.155.2.2	~BrokerInfoMarshaller . . . . .	924
6.155.3	Member Function Documentation . . . . .	924
6.155.3.1	createObject . . . . .	924
6.155.3.2	getDataStructureType . . . . .	925
6.155.3.3	looseMarshal . . . . .	925
6.155.3.4	looseUnmarshal . . . . .	925
6.155.3.5	tightMarshal1 . . . . .	926
6.155.3.6	tightMarshal2 . . . . .	926
6.155.3.7	tightUnmarshal . . . . .	927
6.156	activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller Class	
	Reference . . . . .	927
6.156.1	Detailed Description . . . . .	928
6.156.2	Constructor & Destructor Documentation . . . . .	929
6.156.2.1	BrokerInfoMarshaller . . . . .	929
6.156.2.2	~BrokerInfoMarshaller . . . . .	929
6.156.3	Member Function Documentation . . . . .	929
6.156.3.1	createObject . . . . .	929
6.156.3.2	getDataStructureType . . . . .	929
6.156.3.3	looseMarshal . . . . .	929
6.156.3.4	looseUnmarshal . . . . .	930
6.156.3.5	tightMarshal1 . . . . .	930
6.156.3.6	tightMarshal2 . . . . .	931
6.156.3.7	tightUnmarshal . . . . .	931
6.157	activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller Class	
	Reference . . . . .	932
6.157.1	Detailed Description . . . . .	933

6.157.2 Constructor & Destructor Documentation . . . . .	933
6.157.2.1 BrokerInfoMarshaller . . . . .	933
6.157.2.2 ~BrokerInfoMarshaller . . . . .	933
6.157.3 Member Function Documentation . . . . .	933
6.157.3.1 createObject . . . . .	933
6.157.3.2 getDataStructureType . . . . .	933
6.157.3.3 looseMarshal . . . . .	933
6.157.3.4 looseUnmarshal . . . . .	934
6.157.3.5 tightMarshal1 . . . . .	934
6.157.3.6 tightMarshal2 . . . . .	935
6.157.3.7 tightUnmarshal . . . . .	935
6.158decaf::nio::Buffer Class Reference . . . . .	936
6.158.1 Detailed Description . . . . .	937
6.158.2 Constructor & Destructor Documentation . . . . .	939
6.158.2.1 Buffer . . . . .	939
6.158.2.2 Buffer . . . . .	939
6.158.2.3 ~Buffer . . . . .	939
6.158.3 Member Function Documentation . . . . .	939
6.158.3.1 capacity . . . . .	939
6.158.3.2 clear . . . . .	939
6.158.3.3 flip . . . . .	939
6.158.3.4 hasRemaining . . . . .	940
6.158.3.5 isReadOnly . . . . .	940
6.158.3.6 limit . . . . .	940
6.158.3.7 limit . . . . .	940
6.158.3.8 mark . . . . .	941
6.158.3.9 position . . . . .	941
6.158.3.10position . . . . .	941
6.158.3.11remaining . . . . .	941
6.158.3.12reset . . . . .	942
6.158.3.13rewind . . . . .	942
6.158.4 Field Documentation . . . . .	942
6.158.4.1 _capacity . . . . .	942
6.158.4.2 _limit . . . . .	942
6.158.4.3 _mark . . . . .	942
6.158.4.4 _markSet . . . . .	942
6.158.4.5 _position . . . . .	942
6.159decaf::io::BufferedInputStream Class Reference . . . . .	943
6.159.1 Detailed Description . . . . .	945
6.159.2 Constructor & Destructor Documentation . . . . .	945
6.159.2.1 BufferedInputStream . . . . .	945
6.159.2.2 BufferedInputStream . . . . .	945
6.159.2.3 ~BufferedInputStream . . . . .	946
6.159.3 Member Function Documentation . . . . .	946
6.159.3.1 available . . . . .	946
6.159.3.2 close . . . . .	946
6.159.3.3 doReadArrayBounded . . . . .	946
6.159.3.4 doReadByte . . . . .	946
6.159.3.5 mark . . . . .	947
6.159.3.6 markSupported . . . . .	947

6.159.3.7	reset	947
6.159.3.8	skip	948
6.160	decaf::io::BufferedOutputStream Class Reference	949
6.160.1	Detailed Description	949
6.160.2	Constructor & Destructor Documentation	949
6.160.2.1	BufferedOutputStream	949
6.160.2.2	BufferedOutputStream	950
6.160.2.3	~BufferedOutputStream	950
6.160.3	Member Function Documentation	950
6.160.3.1	doWriteArray	950
6.160.3.2	doWriteArrayBounded	950
6.160.3.3	doWriteByte	950
6.160.3.4	flush	951
6.161	decaf::internal::nio::BufferFactory Class Reference	951
6.161.1	Detailed Description	953
6.161.2	Constructor & Destructor Documentation	954
6.161.2.1	~BufferFactory	954
6.161.3	Member Function Documentation	954
6.161.3.1	createByteBuffer	954
6.161.3.2	createByteBuffer	954
6.161.3.3	createByteBuffer	955
6.161.3.4	createCharBuffer	955
6.161.3.5	createCharBuffer	956
6.161.3.6	createCharBuffer	956
6.161.3.7	createDoubleBuffer	957
6.161.3.8	createDoubleBuffer	957
6.161.3.9	createDoubleBuffer	958
6.161.3.10	createFloatBuffer	958
6.161.3.11	createFloatBuffer	958
6.161.3.12	createFloatBuffer	959
6.161.3.13	createIntBuffer	960
6.161.3.14	createIntBuffer	960
6.161.3.15	createIntBuffer	960
6.161.3.16	createLongBuffer	961
6.161.3.17	createLongBuffer	961
6.161.3.18	createLongBuffer	962
6.161.3.19	createShortBuffer	963
6.161.3.20	createShortBuffer	963
6.161.3.21	createShortBuffer	964
6.162	decaf::nio::BufferOverflowException Class Reference	964
6.162.1	Constructor & Destructor Documentation	965
6.162.1.1	BufferOverflowException	965
6.162.1.2	BufferOverflowException	965
6.162.1.3	BufferOverflowException	965
6.162.1.4	BufferOverflowException	965
6.162.1.5	BufferOverflowException	966
6.162.1.6	BufferOverflowException	966
6.162.1.7	~BufferOverflowException	966
6.162.2	Member Function Documentation	966
6.162.2.1	clone	966

6.163decaf::nio::BufferUnderflowException Class Reference . . . . .	967
6.163.1 Constructor & Destructor Documentation . . . . .	967
6.163.1.1 BufferUnderflowException . . . . .	967
6.163.1.2 BufferUnderflowException . . . . .	968
6.163.1.3 BufferUnderflowException . . . . .	968
6.163.1.4 BufferUnderflowException . . . . .	968
6.163.1.5 BufferUnderflowException . . . . .	968
6.163.1.6 BufferUnderflowException . . . . .	969
6.163.1.7 ~BufferUnderflowException . . . . .	969
6.163.2 Member Function Documentation . . . . .	969
6.163.2.1 clone . . . . .	969
6.164decaf::lang::Byte Class Reference . . . . .	969
6.164.1 Constructor & Destructor Documentation . . . . .	971
6.164.1.1 Byte . . . . .	971
6.164.1.2 Byte . . . . .	972
6.164.1.3 ~Byte . . . . .	972
6.164.2 Member Function Documentation . . . . .	972
6.164.2.1 byteValue . . . . .	972
6.164.2.2 compareTo . . . . .	972
6.164.2.3 compareTo . . . . .	972
6.164.2.4 decode . . . . .	973
6.164.2.5 doubleValue . . . . .	973
6.164.2.6 equals . . . . .	974
6.164.2.7 equals . . . . .	974
6.164.2.8 floatValue . . . . .	974
6.164.2.9 intValue . . . . .	974
6.164.2.10longValue . . . . .	974
6.164.2.11operator< . . . . .	975
6.164.2.12operator< . . . . .	975
6.164.2.13operator== . . . . .	975
6.164.2.14operator== . . . . .	976
6.164.2.15parseByte . . . . .	976
6.164.2.16parseByte . . . . .	976
6.164.2.17shortValue . . . . .	977
6.164.2.18toString . . . . .	977
6.164.2.19toString . . . . .	977
6.164.2.20valueOf . . . . .	977
6.164.2.21valueOf . . . . .	978
6.164.2.22valueOf . . . . .	978
6.164.3 Field Documentation . . . . .	979
6.164.3.1 MAX_VALUE . . . . .	979
6.164.3.2 MIN_VALUE . . . . .	979
6.164.3.3 SIZE . . . . .	979
6.165decaf::internal::util::ByteArrayAdapter Class Reference . . . . .	979
6.165.1 Detailed Description . . . . .	984
6.165.2 Constructor & Destructor Documentation . . . . .	984
6.165.2.1 ByteArrayAdapter . . . . .	984
6.165.2.2 ByteArrayAdapter . . . . .	984
6.165.2.3 ByteArrayAdapter . . . . .	985
6.165.2.4 ByteArrayAdapter . . . . .	985

6.165.2.5	ByteArrayAdapter . . . . .	986
6.165.2.6	ByteArrayAdapter . . . . .	986
6.165.2.7	ByteArrayAdapter . . . . .	987
6.165.2.8	ByteArrayAdapter . . . . .	987
6.165.2.9	~ByteArrayAdapter . . . . .	988
6.165.3	Member Function Documentation . . . . .	988
6.165.3.1	clear . . . . .	988
6.165.3.2	get . . . . .	988
6.165.3.3	getByteArray . . . . .	988
6.165.3.4	getCapacity . . . . .	988
6.165.3.5	getChar . . . . .	989
6.165.3.6	getCharArray . . . . .	989
6.165.3.7	getCharCapacity . . . . .	989
6.165.3.8	getDouble . . . . .	989
6.165.3.9	getDoubleArray . . . . .	990
6.165.3.10	getDoubleAt . . . . .	990
6.165.3.11	getDoubleCapacity . . . . .	990
6.165.3.12	getFloat . . . . .	991
6.165.3.13	getFloatArray . . . . .	991
6.165.3.14	getFloatAt . . . . .	991
6.165.3.15	getFloatCapacity . . . . .	992
6.165.3.16	getInt . . . . .	992
6.165.3.17	getIntArray . . . . .	992
6.165.3.18	getIntAt . . . . .	993
6.165.3.19	getIntCapacity . . . . .	993
6.165.3.20	getLong . . . . .	993
6.165.3.21	getLongArray . . . . .	994
6.165.3.22	getLongAt . . . . .	994
6.165.3.23	getLongCapacity . . . . .	994
6.165.3.24	getShort . . . . .	995
6.165.3.25	getShortArray . . . . .	995
6.165.3.26	getShortAt . . . . .	995
6.165.3.27	getShortCapacity . . . . .	996
6.165.3.28	operator[] . . . . .	996
6.165.3.29	operator[] . . . . .	996
6.165.3.30	put . . . . .	996
6.165.3.31	putChar . . . . .	997
6.165.3.32	putDouble . . . . .	997
6.165.3.33	putDoubleAt . . . . .	998
6.165.3.34	putFloat . . . . .	998
6.165.3.35	putFloatAt . . . . .	999
6.165.3.36	putInt . . . . .	999
6.165.3.37	putIntAt . . . . .	1000
6.165.3.38	putLong . . . . .	1000
6.165.3.39	putLongAt . . . . .	1001
6.165.3.40	putShort . . . . .	1001
6.165.3.41	putShortAt . . . . .	1002
6.165.3.42	read . . . . .	1002
6.165.3.43	resize . . . . .	1003
6.165.3.44	write . . . . .	1003

6.166decaf::internal::nio::ByteBuffer Class Reference . . . . .	1004
6.166.1 Detailed Description . . . . .	1017
6.166.2 Constructor & Destructor Documentation . . . . .	1018
6.166.2.1 ByteBuffer . . . . .	1018
6.166.2.2 ByteBuffer . . . . .	1018
6.166.2.3 ByteBuffer . . . . .	1019
6.166.2.4 ByteBuffer . . . . .	1019
6.166.2.5 ~ByteBuffer . . . . .	1020
6.166.3 Member Function Documentation . . . . .	1020
6.166.3.1 array . . . . .	1020
6.166.3.2 arrayOffset . . . . .	1020
6.166.3.3 asCharBuffer . . . . .	1021
6.166.3.4 asDoubleBuffer . . . . .	1021
6.166.3.5 asFloatBuffer . . . . .	1021
6.166.3.6 asIntBuffer . . . . .	1022
6.166.3.7 asLongBuffer . . . . .	1022
6.166.3.8 asReadOnlyBuffer . . . . .	1023
6.166.3.9 asShortBuffer . . . . .	1023
6.166.3.10compact . . . . .	1023
6.166.3.11duplicate . . . . .	1024
6.166.3.12get . . . . .	1024
6.166.3.13get . . . . .	1025
6.166.3.14getChar . . . . .	1025
6.166.3.15getChar . . . . .	1025
6.166.3.16getDouble . . . . .	1026
6.166.3.17getDouble . . . . .	1026
6.166.3.18getFloat . . . . .	1027
6.166.3.19getFloat . . . . .	1027
6.166.3.20getInt . . . . .	1027
6.166.3.21getInt . . . . .	1028
6.166.3.22getLong . . . . .	1028
6.166.3.23getLong . . . . .	1029
6.166.3.24getShort . . . . .	1029
6.166.3.25getShort . . . . .	1029
6.166.3.26hasArray . . . . .	1030
6.166.3.27isReadOnly . . . . .	1030
6.166.3.28put . . . . .	1030
6.166.3.29put . . . . .	1031
6.166.3.30putChar . . . . .	1031
6.166.3.31putChar . . . . .	1032
6.166.3.32putDouble . . . . .	1032
6.166.3.33putDouble . . . . .	1033
6.166.3.34putFloat . . . . .	1033
6.166.3.35putFloat . . . . .	1034
6.166.3.36putInt . . . . .	1034
6.166.3.37putInt . . . . .	1035
6.166.3.38putLong . . . . .	1035
6.166.3.39putLong . . . . .	1036
6.166.3.40putShort . . . . .	1036
6.166.3.41putShort . . . . .	1037



6.166.3.42	setReadOnly	1037
6.166.3.43	slice	1038
6.167	decaf::io::ByteArrayInputStream Class Reference	1038
6.167.1	Detailed Description	1041
6.167.2	Constructor & Destructor Documentation	1041
6.167.2.1	ByteArrayInputStream	1041
6.167.2.2	ByteArrayInputStream	1041
6.167.2.3	ByteArrayInputStream	1042
6.167.2.4	ByteArrayInputStream	1042
6.167.2.5	~ByteArrayInputStream	1043
6.167.3	Member Function Documentation	1043
6.167.3.1	available	1043
6.167.3.2	doReadArrayBounded	1043
6.167.3.3	doReadByte	1043
6.167.3.4	mark	1043
6.167.3.5	markSupported	1044
6.167.3.6	reset	1044
6.167.3.7	setByteArray	1045
6.167.3.8	setByteArray	1045
6.167.3.9	setByteArray	1045
6.167.3.10	skip	1046
6.168	decaf::io::ByteArrayOutputStream Class Reference	1046
6.168.1	Constructor & Destructor Documentation	1047
6.168.1.1	ByteArrayOutputStream	1047
6.168.1.2	ByteArrayOutputStream	1048
6.168.1.3	~ByteArrayOutputStream	1048
6.168.2	Member Function Documentation	1048
6.168.2.1	doWriteArrayBounded	1048
6.168.2.2	doWriteByte	1048
6.168.2.3	reset	1048
6.168.2.4	size	1048
6.168.2.5	toArray	1049
6.168.2.6	toString	1049
6.168.2.7	writeTo	1049
6.169	decaf::nio::ByteBuffer Class Reference	1049
6.169.1	Detailed Description	1054
6.169.2	Constructor & Destructor Documentation	1055
6.169.2.1	ByteBuffer	1055
6.169.2.2	~ByteBuffer	1056
6.169.3	Member Function Documentation	1056
6.169.3.1	allocate	1056
6.169.3.2	array	1056
6.169.3.3	arrayOffset	1057
6.169.3.4	asCharBuffer	1057
6.169.3.5	asDoubleBuffer	1058
6.169.3.6	asFloatBuffer	1058
6.169.3.7	asIntBuffer	1058
6.169.3.8	asLongBuffer	1059
6.169.3.9	asReadOnlyBuffer	1059
6.169.3.10	asShortBuffer	1059

6.169.3.11compact	1060
6.169.3.12compareTo	1060
6.169.3.13duplicate	1060
6.169.3.14equals	1061
6.169.3.15get	1061
6.169.3.16get	1061
6.169.3.17get	1062
6.169.3.18get	1062
6.169.3.19getChar	1063
6.169.3.20getChar	1063
6.169.3.21getDouble	1064
6.169.3.22getDouble	1064
6.169.3.23getFloat	1064
6.169.3.24getFloat	1065
6.169.3.25getInt	1065
6.169.3.26getInt	1066
6.169.3.27getLong	1066
6.169.3.28getLong	1066
6.169.3.29getShort	1067
6.169.3.30getShort	1067
6.169.3.31hasArray	1068
6.169.3.32isReadOnly	1068
6.169.3.33operator<	1068
6.169.3.34operator==	1068
6.169.3.35put	1068
6.169.3.36put	1069
6.169.3.37put	1069
6.169.3.38put	1070
6.169.3.39put	1071
6.169.3.40putChar	1071
6.169.3.41putChar	1072
6.169.3.42putDouble	1072
6.169.3.43putDouble	1073
6.169.3.44putFloat	1073
6.169.3.45putFloat	1074
6.169.3.46putInt	1074
6.169.3.47putInt	1075
6.169.3.48putLong	1075
6.169.3.49putLong	1076
6.169.3.50putShort	1076
6.169.3.51putShort	1077
6.169.3.52slice	1077
6.169.3.53toString	1078
6.169.3.54wrap	1078
6.169.3.55wrap	1078
6.170cms::BytesMessage Class Reference	1079
6.170.1 Detailed Description	1082
6.170.2 Constructor & Destructor Documentation	1083
6.170.2.1 ~BytesMessage	1083
6.170.3 Member Function Documentation	1083

6.170.3.1	clone	1083
6.170.3.2	getBodyBytes	1083
6.170.3.3	getBodyLength	1084
6.170.3.4	readBoolean	1084
6.170.3.5	readByte	1085
6.170.3.6	readBytes	1085
6.170.3.7	readBytes	1086
6.170.3.8	readChar	1087
6.170.3.9	readDouble	1087
6.170.3.10	readFloat	1088
6.170.3.11	readInt	1088
6.170.3.12	readLong	1089
6.170.3.13	readShort	1089
6.170.3.14	readString	1090
6.170.3.15	readUnsignedShort	1090
6.170.3.16	readUTF	1091
6.170.3.17	reset	1091
6.170.3.18	setBodyBytes	1091
6.170.3.19	writeBoolean	1092
6.170.3.20	writeByte	1092
6.170.3.21	writeBytes	1093
6.170.3.22	writeBytes	1093
6.170.3.23	writeChar	1094
6.170.3.24	writeDouble	1094
6.170.3.25	writeFloat	1094
6.170.3.26	writeInt	1095
6.170.3.27	writeLong	1095
6.170.3.28	writeShort	1096
6.170.3.29	writeString	1096
6.170.3.30	writeUnsignedShort	1096
6.170.3.31	writeUTF	1097
6.171	activemq::cmsutil::CachedConsumer Class Reference	1097
6.171.1	Detailed Description	1098
6.171.2	Constructor & Destructor Documentation	1099
6.171.2.1	CachedConsumer	1099
6.171.2.2	CachedConsumer	1099
6.171.2.3	~CachedConsumer	1099
6.171.3	Member Function Documentation	1099
6.171.3.1	close	1099
6.171.3.2	getMessageListener	1099
6.171.3.3	getMessageSelector	1099
6.171.3.4	operator=	1100
6.171.3.5	receive	1100
6.171.3.6	receive	1100
6.171.3.7	receiveNoWait	1100
6.171.3.8	setMessageListener	1101
6.172	activemq::cmsutil::CachedProducer Class Reference	1101
6.172.1	Detailed Description	1103
6.172.2	Constructor & Destructor Documentation	1103
6.172.2.1	CachedProducer	1103

6.172.2.2	CachedProducer . . . . .	1103
6.172.2.3	~CachedProducer . . . . .	1103
6.172.3	Member Function Documentation . . . . .	1103
6.172.3.1	close . . . . .	1103
6.172.3.2	getDeliveryMode . . . . .	1103
6.172.3.3	getDisableMessageID . . . . .	1103
6.172.3.4	getDisableMessageTimeStamp . . . . .	1104
6.172.3.5	getPriority . . . . .	1104
6.172.3.6	getTimeToLive . . . . .	1104
6.172.3.7	operator= . . . . .	1105
6.172.3.8	send . . . . .	1105
6.172.3.9	send . . . . .	1105
6.172.3.10	send . . . . .	1106
6.172.3.11	send . . . . .	1106
6.172.3.12	setDeliveryMode . . . . .	1107
6.172.3.13	setDisableMessageID . . . . .	1107
6.172.3.14	setDisableMessageTimeStamp . . . . .	1107
6.172.3.15	setPriority . . . . .	1108
6.172.3.16	setTimeToLive . . . . .	1108
6.173	decaf::util::concurrent::Callable< V > Class Template Reference . . . . .	1108
6.173.1	Detailed Description . . . . .	1109
6.173.2	Constructor & Destructor Documentation . . . . .	1109
6.173.2.1	~Callable . . . . .	1109
6.173.3	Member Function Documentation . . . . .	1109
6.173.3.1	call . . . . .	1109
6.174	decaf::util::concurrent::CancellationException Class Reference . . . . .	1110
6.174.1	Constructor & Destructor Documentation . . . . .	1110
6.174.1.1	CancellationException . . . . .	1110
6.174.1.2	CancellationException . . . . .	1110
6.174.1.3	CancellationException . . . . .	1111
6.174.1.4	CancellationException . . . . .	1111
6.174.1.5	CancellationException . . . . .	1111
6.174.1.6	CancellationException . . . . .	1111
6.174.1.7	~CancellationException . . . . .	1112
6.174.2	Member Function Documentation . . . . .	1112
6.174.2.1	clone . . . . .	1112
6.175	decaf::security::cert::Certificate Class Reference . . . . .	1112
6.175.1	Detailed Description . . . . .	1113
6.175.2	Constructor & Destructor Documentation . . . . .	1113
6.175.2.1	~Certificate . . . . .	1113
6.175.3	Member Function Documentation . . . . .	1113
6.175.3.1	equals . . . . .	1113
6.175.3.2	getEncoded . . . . .	1114
6.175.3.3	getPublicKey . . . . .	1114
6.175.3.4	getPublicKey . . . . .	1114
6.175.3.5	getType . . . . .	1114
6.175.3.6	toString . . . . .	1115
6.175.3.7	verify . . . . .	1115
6.175.3.8	verify . . . . .	1115
6.176	decaf::security::cert::CertificateEncodingException Class Reference . . . . .	1116

6.176.1 Constructor & Destructor Documentation . . . . .	1117
6.176.1.1 CertificateEncodingException . . . . .	1117
6.176.1.2 CertificateEncodingException . . . . .	1117
6.176.1.3 CertificateEncodingException . . . . .	1117
6.176.1.4 CertificateEncodingException . . . . .	1117
6.176.1.5 ~CertificateEncodingException . . . . .	1118
6.176.2 Member Function Documentation . . . . .	1118
6.176.2.1 clone . . . . .	1118
6.177decaf::security::cert::CertificateException Class Reference . . . . .	1118
6.177.1 Constructor & Destructor Documentation . . . . .	1119
6.177.1.1 CertificateException . . . . .	1119
6.177.1.2 CertificateException . . . . .	1119
6.177.1.3 CertificateException . . . . .	1119
6.177.1.4 CertificateException . . . . .	1119
6.177.1.5 ~CertificateException . . . . .	1120
6.177.2 Member Function Documentation . . . . .	1120
6.177.2.1 clone . . . . .	1120
6.178decaf::security::cert::CertificateExpiredException Class Reference . . . . .	1120
6.178.1 Constructor & Destructor Documentation . . . . .	1121
6.178.1.1 CertificateExpiredException . . . . .	1121
6.178.1.2 CertificateExpiredException . . . . .	1121
6.178.1.3 CertificateExpiredException . . . . .	1121
6.178.1.4 CertificateExpiredException . . . . .	1121
6.178.1.5 ~CertificateExpiredException . . . . .	1122
6.178.2 Member Function Documentation . . . . .	1122
6.178.2.1 clone . . . . .	1122
6.179decaf::security::cert::CertificateNotYetValidException Class Reference . . . . .	1122
6.179.1 Constructor & Destructor Documentation . . . . .	1123
6.179.1.1 CertificateNotYetValidException . . . . .	1123
6.179.1.2 CertificateNotYetValidException . . . . .	1123
6.179.1.3 CertificateNotYetValidException . . . . .	1123
6.179.1.4 CertificateNotYetValidException . . . . .	1123
6.179.1.5 ~CertificateNotYetValidException . . . . .	1124
6.179.2 Member Function Documentation . . . . .	1124
6.179.2.1 clone . . . . .	1124
6.180decaf::security::cert::CertificateParsingException Class Reference . . . . .	1124
6.180.1 Constructor & Destructor Documentation . . . . .	1125
6.180.1.1 CertificateParsingException . . . . .	1125
6.180.1.2 CertificateParsingException . . . . .	1125
6.180.1.3 CertificateParsingException . . . . .	1125
6.180.1.4 CertificateParsingException . . . . .	1125
6.180.1.5 ~CertificateParsingException . . . . .	1126
6.180.2 Member Function Documentation . . . . .	1126
6.180.2.1 clone . . . . .	1126
6.181decaf::lang::Character Class Reference . . . . .	1126
6.181.1 Constructor & Destructor Documentation . . . . .	1128
6.181.1.1 Character . . . . .	1128
6.181.2 Member Function Documentation . . . . .	1129
6.181.2.1 byteValue . . . . .	1129
6.181.2.2 compareTo . . . . .	1129

6.181.2.3	compareTo	1129
6.181.2.4	digit	1130
6.181.2.5	doubleValue	1130
6.181.2.6	equals	1130
6.181.2.7	equals	1130
6.181.2.8	floatValue	1131
6.181.2.9	intValue	1131
6.181.2.10	isDigit	1131
6.181.2.11	isISOControl	1131
6.181.2.12	isLetter	1131
6.181.2.13	isLetterOrDigit	1131
6.181.2.14	isLowerCase	1132
6.181.2.15	isUpperCase	1132
6.181.2.16	isWhitespace	1132
6.181.2.17	longValue	1132
6.181.2.18	operator<	1132
6.181.2.19	operator<	1133
6.181.2.20	operator==	1133
6.181.2.21	operator==	1133
6.181.2.22	shortValue	1134
6.181.2.23	toString	1134
6.181.2.24	valueOf	1134
6.181.3	Field Documentation	1134
6.181.3.1	MAX_RADIX	1134
6.181.3.2	MAX_VALUE	1134
6.181.3.3	MIN_RADIX	1134
6.181.3.4	MIN_VALUE	1135
6.181.3.5	SIZE	1135
6.182	decaf::internal::nio::CharArrayBuffer Class Reference	1135
6.182.1	Constructor & Destructor Documentation	1140
6.182.1.1	CharArrayBuffer	1140
6.182.1.2	CharArrayBuffer	1140
6.182.1.3	CharArrayBuffer	1141
6.182.1.4	CharArrayBuffer	1141
6.182.1.5	~CharArrayBuffer	1141
6.182.2	Member Function Documentation	1141
6.182.2.1	array	1141
6.182.2.2	arrayOffset	1142
6.182.2.3	asReadOnlyBuffer	1142
6.182.2.4	compact	1143
6.182.2.5	duplicate	1143
6.182.2.6	get	1144
6.182.2.7	get	1144
6.182.2.8	hasArray	1145
6.182.2.9	isReadOnly	1145
6.182.2.10	put	1145
6.182.2.11	put	1146
6.182.2.12	setReadOnly	1146
6.182.2.13	slice	1146
6.182.2.14	subSequence	1147

6.182.3 Field Documentation . . . . .	1148
6.182.3.1 _array . . . . .	1148
6.182.3.2 length . . . . .	1148
6.182.3.3 offset . . . . .	1148
6.182.3.4 readOnly . . . . .	1148
6.183decaf::nio::CharBuffer Class Reference . . . . .	1148
6.183.1 Detailed Description . . . . .	1151
6.183.2 Constructor & Destructor Documentation . . . . .	1152
6.183.2.1 CharBuffer . . . . .	1152
6.183.2.2 ~CharBuffer . . . . .	1152
6.183.3 Member Function Documentation . . . . .	1152
6.183.3.1 allocate . . . . .	1152
6.183.3.2 append . . . . .	1153
6.183.3.3 append . . . . .	1153
6.183.3.4 append . . . . .	1154
6.183.3.5 array . . . . .	1154
6.183.3.6 arrayOffset . . . . .	1155
6.183.3.7 asReadOnlyBuffer . . . . .	1155
6.183.3.8 charAt . . . . .	1156
6.183.3.9 compact . . . . .	1156
6.183.3.10compareTo . . . . .	1157
6.183.3.11duplicate . . . . .	1157
6.183.3.12equals . . . . .	1157
6.183.3.13get . . . . .	1157
6.183.3.14get . . . . .	1157
6.183.3.15get . . . . .	1158
6.183.3.16get . . . . .	1158
6.183.3.17hasArray . . . . .	1159
6.183.3.18length . . . . .	1159
6.183.3.19operator< . . . . .	1160
6.183.3.20operator== . . . . .	1160
6.183.3.21put . . . . .	1160
6.183.3.22put . . . . .	1160
6.183.3.23put . . . . .	1161
6.183.3.24put . . . . .	1162
6.183.3.25put . . . . .	1162
6.183.3.26put . . . . .	1163
6.183.3.27put . . . . .	1163
6.183.3.28read . . . . .	1164
6.183.3.29slice . . . . .	1164
6.183.3.30subSequence . . . . .	1165
6.183.3.31toString . . . . .	1165
6.183.3.32wrap . . . . .	1166
6.183.3.33wrap . . . . .	1166
6.184decaf::lang::CharSequence Class Reference . . . . .	1167
6.184.1 Detailed Description . . . . .	1167
6.184.2 Constructor & Destructor Documentation . . . . .	1168
6.184.2.1 ~CharSequence . . . . .	1168
6.184.3 Member Function Documentation . . . . .	1168
6.184.3.1 charAt . . . . .	1168

6.184.3.2	length	1168
6.184.3.3	subSequence	1168
6.184.3.4	toString	1169
6.185	decaf::util::zip::CheckedInputStream Class Reference	1169
6.185.1	Detailed Description	1170
6.185.2	Constructor & Destructor Documentation	1170
6.185.2.1	CheckedInputStream	1170
6.185.2.2	~CheckedInputStream	1171
6.185.3	Member Function Documentation	1171
6.185.3.1	doReadArrayBounded	1171
6.185.3.2	doReadByte	1171
6.185.3.3	getChecksum	1171
6.185.3.4	skip	1171
6.186	decaf::util::zip::CheckedOutputStream Class Reference	1172
6.186.1	Detailed Description	1173
6.186.2	Constructor & Destructor Documentation	1173
6.186.2.1	CheckedOutputStream	1173
6.186.2.2	~CheckedOutputStream	1173
6.186.3	Member Function Documentation	1173
6.186.3.1	doWriteArrayBounded	1173
6.186.3.2	doWriteByte	1173
6.186.3.3	getChecksum	1174
6.187	decaf::util::zip::Checksum Class Reference	1174
6.187.1	Detailed Description	1175
6.187.2	Constructor & Destructor Documentation	1175
6.187.2.1	~Checksum	1175
6.187.3	Member Function Documentation	1175
6.187.3.1	getValue	1175
6.187.3.2	reset	1175
6.187.3.3	update	1175
6.187.3.4	update	1176
6.187.3.5	update	1176
6.187.3.6	update	1176
6.188	decaf::lang::exceptions::ClassCastException Class Reference	1177
6.188.1	Constructor & Destructor Documentation	1177
6.188.1.1	ClassCastException	1177
6.188.1.2	ClassCastException	1177
6.188.1.3	ClassCastException	1178
6.188.1.4	ClassCastException	1178
6.188.1.5	ClassCastException	1178
6.188.1.6	ClassCastException	1178
6.188.1.7	~ClassCastException	1179
6.188.2	Member Function Documentation	1179
6.188.2.1	clone	1179
6.189	cms::Closeable Class Reference	1179
6.189.1	Detailed Description	1180
6.189.2	Constructor & Destructor Documentation	1180
6.189.2.1	~Closeable	1180
6.189.3	Member Function Documentation	1180
6.189.3.1	close	1180



6.190	decaf::io::Closeable Class Reference . . . . .	1180
6.190.1	Detailed Description . . . . .	1181
6.190.2	Constructor & Destructor Documentation . . . . .	1181
6.190.2.1	~Closeable . . . . .	1181
6.190.3	Member Function Documentation . . . . .	1181
6.190.3.1	close . . . . .	1181
6.191	activemq::transport::failover::CloseTransportsTask Class Reference . . . . .	1182
6.191.1	Constructor & Destructor Documentation . . . . .	1182
6.191.1.1	CloseTransportsTask . . . . .	1182
6.191.1.2	~CloseTransportsTask . . . . .	1182
6.191.2	Member Function Documentation . . . . .	1182
6.191.2.1	add . . . . .	1182
6.191.2.2	isPending . . . . .	1182
6.191.2.3	iterate . . . . .	1183
6.192	activemq::cmsutil::CmsAccessor Class Reference . . . . .	1183
6.192.1	Detailed Description . . . . .	1184
6.192.2	Constructor & Destructor Documentation . . . . .	1185
6.192.2.1	CmsAccessor . . . . .	1185
6.192.2.2	CmsAccessor . . . . .	1185
6.192.2.3	~CmsAccessor . . . . .	1185
6.192.3	Member Function Documentation . . . . .	1185
6.192.3.1	checkConnectionFactory . . . . .	1185
6.192.3.2	createConnection . . . . .	1185
6.192.3.3	createSession . . . . .	1185
6.192.3.4	destroy . . . . .	1186
6.192.3.5	getConnectionFactory . . . . .	1186
6.192.3.6	getConnectionFactory . . . . .	1186
6.192.3.7	getResourceLifecycleManager . . . . .	1186
6.192.3.8	getResourceLifecycleManager . . . . .	1186
6.192.3.9	getSessionAcknowledgeMode . . . . .	1186
6.192.3.10	init . . . . .	1187
6.192.3.11	operator= . . . . .	1187
6.192.3.12	setConnectionFactory . . . . .	1187
6.192.3.13	setSessionAcknowledgeMode . . . . .	1187
6.193	activemq::cmsutil::CmsDestinationAccessor Class Reference . . . . .	1187
6.193.1	Detailed Description . . . . .	1188
6.193.2	Constructor & Destructor Documentation . . . . .	1189
6.193.2.1	CmsDestinationAccessor . . . . .	1189
6.193.2.2	CmsDestinationAccessor . . . . .	1189
6.193.2.3	~CmsDestinationAccessor . . . . .	1189
6.193.3	Member Function Documentation . . . . .	1189
6.193.3.1	checkDestinationResolver . . . . .	1189
6.193.3.2	destroy . . . . .	1189
6.193.3.3	getDestinationResolver . . . . .	1189
6.193.3.4	getDestinationResolver . . . . .	1189
6.193.3.5	init . . . . .	1189
6.193.3.6	isPubSubDomain . . . . .	1190
6.193.3.7	operator= . . . . .	1190
6.193.3.8	resolveDestinationName . . . . .	1190
6.193.3.9	setDestinationResolver . . . . .	1190

6.193.3.10setPubSubDomain . . . . .	1190
6.194cms::CMSException Class Reference . . . . .	1190
6.194.1 Detailed Description . . . . .	1192
6.194.2 Constructor & Destructor Documentation . . . . .	1192
6.194.2.1 CMSException . . . . .	1192
6.194.2.2 CMSException . . . . .	1192
6.194.2.3 CMSException . . . . .	1192
6.194.2.4 CMSException . . . . .	1192
6.194.2.5 ~CMSException . . . . .	1192
6.194.3 Member Function Documentation . . . . .	1192
6.194.3.1 getCause . . . . .	1192
6.194.3.2 getMessage . . . . .	1193
6.194.3.3 getStackTrace . . . . .	1193
6.194.3.4 getStackTraceString . . . . .	1193
6.194.3.5 printStackTrace . . . . .	1193
6.194.3.6 printStackTrace . . . . .	1193
6.194.3.7 setMark . . . . .	1193
6.194.3.8 what . . . . .	1194
6.195activemq::util::CMSExceptionSupport Class Reference . . . . .	1194
6.195.1 Constructor & Destructor Documentation . . . . .	1195
6.195.1.1 ~CMSExceptionSupport . . . . .	1195
6.195.2 Member Function Documentation . . . . .	1195
6.195.2.1 create . . . . .	1195
6.195.2.2 create . . . . .	1195
6.195.2.3 createMessageEOFException . . . . .	1195
6.195.2.4 createMessageFormatException . . . . .	1195
6.196cms::CMSProperties Class Reference . . . . .	1195
6.196.1 Detailed Description . . . . .	1196
6.196.2 Constructor & Destructor Documentation . . . . .	1197
6.196.2.1 ~CMSProperties . . . . .	1197
6.196.3 Member Function Documentation . . . . .	1197
6.196.3.1 clear . . . . .	1197
6.196.3.2 clone . . . . .	1197
6.196.3.3 copy . . . . .	1197
6.196.3.4 getProperty . . . . .	1197
6.196.3.5 getProperty . . . . .	1198
6.196.3.6 hasProperty . . . . .	1198
6.196.3.7 isEmpty . . . . .	1198
6.196.3.8 remove . . . . .	1198
6.196.3.9 setProperty . . . . .	1199
6.196.3.10toArray . . . . .	1199
6.196.3.11toString . . . . .	1199
6.197cms::CMSSecurityException Class Reference . . . . .	1199
6.197.1 Detailed Description . . . . .	1200
6.197.2 Constructor & Destructor Documentation . . . . .	1200
6.197.2.1 CMSSecurityException . . . . .	1200
6.197.2.2 CMSSecurityException . . . . .	1200
6.197.2.3 CMSSecurityException . . . . .	1200
6.197.2.4 CMSSecurityException . . . . .	1200
6.197.2.5 ~CMSSecurityException . . . . .	1200

6.198activemq::cmsutil::CmsTemplate Class Reference . . . . .	1201
6.198.1 Detailed Description . . . . .	1204
6.198.2 Constructor & Destructor Documentation . . . . .	1205
6.198.2.1 CmsTemplate . . . . .	1205
6.198.2.2 CmsTemplate . . . . .	1205
6.198.2.3 CmsTemplate . . . . .	1205
6.198.2.4 ~CmsTemplate . . . . .	1205
6.198.3 Member Function Documentation . . . . .	1205
6.198.3.1 destroy . . . . .	1205
6.198.3.2 execute . . . . .	1205
6.198.3.3 execute . . . . .	1205
6.198.3.4 execute . . . . .	1206
6.198.3.5 execute . . . . .	1206
6.198.3.6 getDefaultDestination . . . . .	1206
6.198.3.7 getDefaultDestination . . . . .	1207
6.198.3.8 getDefaultDestinationName . . . . .	1207
6.198.3.9 getDeliveryMode . . . . .	1207
6.198.3.10getPriority . . . . .	1207
6.198.3.11getReceiveTimeout . . . . .	1207
6.198.3.12getTimeToLive . . . . .	1207
6.198.3.13init . . . . .	1207
6.198.3.14isExplicitQosEnabled . . . . .	1208
6.198.3.15isMessageIdEnabled . . . . .	1208
6.198.3.16isMessageTimestampEnabled . . . . .	1208
6.198.3.17isNoLocal . . . . .	1208
6.198.3.18operator= . . . . .	1208
6.198.3.19receive . . . . .	1208
6.198.3.20receive . . . . .	1209
6.198.3.21receive . . . . .	1209
6.198.3.22receiveSelected . . . . .	1209
6.198.3.23receiveSelected . . . . .	1210
6.198.3.24receiveSelected . . . . .	1210
6.198.3.25send . . . . .	1211
6.198.3.26send . . . . .	1211
6.198.3.27send . . . . .	1211
6.198.3.28setDefaultDestination . . . . .	1212
6.198.3.29setDefaultDestinationName . . . . .	1212
6.198.3.30setDeliveryMode . . . . .	1212
6.198.3.31setDeliveryPersistent . . . . .	1213
6.198.3.32setExplicitQosEnabled . . . . .	1213
6.198.3.33setMessageIdEnabled . . . . .	1213
6.198.3.34setMessageTimestampEnabled . . . . .	1213
6.198.3.35setNoLocal . . . . .	1213
6.198.3.36setPriority . . . . .	1213
6.198.3.37setPubSubDomain . . . . .	1214
6.198.3.38setReceiveTimeout . . . . .	1214
6.198.3.39setTimeToLive . . . . .	1214
6.198.4 Friends And Related Function Documentation . . . . .	1215
6.198.4.1 ProducerExecutor . . . . .	1215
6.198.4.2 ReceiveExecutor . . . . .	1215

6.198.4.3 ResolveProducerExecutor . . . . .	1215
6.198.4.4 ResolveReceiveExecutor . . . . .	1215
6.198.4.5 SendExecutor . . . . .	1215
6.198.5 Field Documentation . . . . .	1215
6.198.5.1 DEFAULT_PRIORITY . . . . .	1215
6.198.5.2 DEFAULT_TIME_TO_LIVE . . . . .	1215
6.198.5.3 RECEIVE_TIMEOUT_INDEFINITE_WAIT . . . . .	1215
6.198.5.4 RECEIVE_TIMEOUT_NO_WAIT . . . . .	1215
6.199code Struct Reference . . . . .	1215
6.199.1 Field Documentation . . . . .	1216
6.199.1.1 bits . . . . .	1216
6.199.1.2 op . . . . .	1216
6.199.1.3 val . . . . .	1216
6.200decaf::util::Collection< E > Class Template Reference . . . . .	1216
6.200.1 Detailed Description . . . . .	1217
6.200.2 Constructor & Destructor Documentation . . . . .	1218
6.200.2.1 ~Collection . . . . .	1218
6.200.3 Member Function Documentation . . . . .	1218
6.200.3.1 add . . . . .	1218
6.200.3.2 addAll . . . . .	1219
6.200.3.3 clear . . . . .	1220
6.200.3.4 contains . . . . .	1221
6.200.3.5 containsAll . . . . .	1222
6.200.3.6 equals . . . . .	1222
6.200.3.7 isEmpty . . . . .	1223
6.200.3.8 remove . . . . .	1223
6.200.3.9 removeAll . . . . .	1224
6.200.3.10retainAll . . . . .	1225
6.200.3.11size . . . . .	1226
6.200.3.12toArray . . . . .	1226
6.201activemq::commands::Command Class Reference . . . . .	1227
6.201.1 Constructor & Destructor Documentation . . . . .	1228
6.201.1.1 ~Command . . . . .	1228
6.201.2 Member Function Documentation . . . . .	1228
6.201.2.1 getCommandId . . . . .	1228
6.201.2.2 isBrokerInfo . . . . .	1228
6.201.2.3 isConnectionInfo . . . . .	1229
6.201.2.4 isConsumerInfo . . . . .	1229
6.201.2.5 isKeepAliveInfo . . . . .	1229
6.201.2.6 isMessage . . . . .	1229
6.201.2.7 isMessageAck . . . . .	1229
6.201.2.8 isMessageDispatch . . . . .	1229
6.201.2.9 isMessageDispatchNotification . . . . .	1229
6.201.2.10isProducerAck . . . . .	1230
6.201.2.11isProducerInfo . . . . .	1230
6.201.2.12isRemoveInfo . . . . .	1230
6.201.2.13isRemoveSubscriptionInfo . . . . .	1230
6.201.2.14isResponse . . . . .	1230
6.201.2.15isResponseRequired . . . . .	1230
6.201.2.16isShutdownInfo . . . . .	1230

6.201.2.17	sTransactionInfo . . . . .	1231
6.201.2.18	sWireFormatInfo . . . . .	1231
6.201.2.19	setCommandId . . . . .	1231
6.201.2.20	setResponseRequired . . . . .	1231
6.201.2.21	toString . . . . .	1231
6.201.2.22	visit . . . . .	1232
6.202	activemq::state::CommandVisitor Class Reference . . . . .	1233
6.202.1	Detailed Description . . . . .	1235
6.202.2	Constructor & Destructor Documentation . . . . .	1235
6.202.2.1	~CommandVisitor . . . . .	1235
6.202.3	Member Function Documentation . . . . .	1235
6.202.3.1	processBeginTransaction . . . . .	1235
6.202.3.2	processBrokerError . . . . .	1236
6.202.3.3	processBrokerInfo . . . . .	1236
6.202.3.4	processCommitTransactionOnePhase . . . . .	1236
6.202.3.5	processCommitTransactionTwoPhase . . . . .	1236
6.202.3.6	processConnectionControl . . . . .	1236
6.202.3.7	processConnectionError . . . . .	1236
6.202.3.8	processConnectionInfo . . . . .	1236
6.202.3.9	processConsumerControl . . . . .	1237
6.202.3.10	processConsumerInfo . . . . .	1237
6.202.3.11	processControlCommand . . . . .	1237
6.202.3.12	processDestinationInfo . . . . .	1237
6.202.3.13	processEndTransaction . . . . .	1237
6.202.3.14	processFlushCommand . . . . .	1238
6.202.3.15	processForgetTransaction . . . . .	1238
6.202.3.16	processKeepAliveInfo . . . . .	1238
6.202.3.17	processMessage . . . . .	1238
6.202.3.18	processMessageAck . . . . .	1238
6.202.3.19	processMessageDispatch . . . . .	1239
6.202.3.20	processMessageDispatchNotification . . . . .	1239
6.202.3.21	processMessagePull . . . . .	1239
6.202.3.22	processPrepareTransaction . . . . .	1239
6.202.3.23	processProducerAck . . . . .	1239
6.202.3.24	processProducerInfo . . . . .	1239
6.202.3.25	processRecoverTransactions . . . . .	1239
6.202.3.26	processRemoveConnection . . . . .	1239
6.202.3.27	processRemoveConsumer . . . . .	1240
6.202.3.28	processRemoveDestination . . . . .	1240
6.202.3.29	processRemoveInfo . . . . .	1240
6.202.3.30	processRemoveProducer . . . . .	1240
6.202.3.31	processRemoveSession . . . . .	1240
6.202.3.32	processRemoveSubscriptionInfo . . . . .	1241
6.202.3.33	processReplayCommand . . . . .	1241
6.202.3.34	processResponse . . . . .	1241
6.202.3.35	processRollbackTransaction . . . . .	1241
6.202.3.36	processSessionInfo . . . . .	1241
6.202.3.37	processShutdownInfo . . . . .	1241
6.202.3.38	processTransactionInfo . . . . .	1241
6.202.3.39	processWireFormat . . . . .	1241

6.203	activemq::state::CommandVisitorAdapter Class Reference . . . . .	1242
6.203.1	Detailed Description . . . . .	1244
6.203.2	Constructor & Destructor Documentation . . . . .	1245
6.203.2.1	~CommandVisitorAdapter . . . . .	1245
6.203.3	Member Function Documentation . . . . .	1245
6.203.3.1	processBeginTransaction . . . . .	1245
6.203.3.2	processBrokerError . . . . .	1245
6.203.3.3	processBrokerInfo . . . . .	1245
6.203.3.4	processCommitTransactionOnePhase . . . . .	1245
6.203.3.5	processCommitTransactionTwoPhase . . . . .	1245
6.203.3.6	processConnectionControl . . . . .	1245
6.203.3.7	processConnectionError . . . . .	1245
6.203.3.8	processConnectionInfo . . . . .	1245
6.203.3.9	processConsumerControl . . . . .	1245
6.203.3.10	processConsumerInfo . . . . .	1245
6.203.3.11	processControlCommand . . . . .	1245
6.203.3.12	processDestinationInfo . . . . .	1245
6.203.3.13	processEndTransaction . . . . .	1245
6.203.3.14	processFlushCommand . . . . .	1245
6.203.3.15	processForgetTransaction . . . . .	1245
6.203.3.16	processKeepAliveInfo . . . . .	1245
6.203.3.17	processMessage . . . . .	1245
6.203.3.18	processMessageAck . . . . .	1245
6.203.3.19	processMessageDispatch . . . . .	1245
6.203.3.20	processMessageDispatchNotification . . . . .	1245
6.203.3.21	processMessagePull . . . . .	1245
6.203.3.22	processPrepareTransaction . . . . .	1245
6.203.3.23	processProducerAck . . . . .	1245
6.203.3.24	processProducerInfo . . . . .	1245
6.203.3.25	processRecoverTransactions . . . . .	1245
6.203.3.26	processRemoveConnection . . . . .	1245
6.203.3.27	processRemoveConsumer . . . . .	1245
6.203.3.28	processRemoveDestination . . . . .	1245
6.203.3.29	processRemoveInfo . . . . .	1245
6.203.3.30	processRemoveProducer . . . . .	1247
6.203.3.31	processRemoveSession . . . . .	1247
6.203.3.32	processRemoveSubscriptionInfo . . . . .	1247
6.203.3.33	processReplayCommand . . . . .	1247
6.203.3.34	processResponse . . . . .	1247
6.203.3.35	processRollbackTransaction . . . . .	1247
6.203.3.36	processSessionInfo . . . . .	1247
6.203.3.37	processShutdownInfo . . . . .	1247
6.203.3.38	processTransactionInfo . . . . .	1247
6.203.3.39	processWireFormat . . . . .	1248
6.204	decaf::lang::Comparable< T > Class Template Reference . . . . .	1248
6.204.1	Detailed Description . . . . .	1248
6.204.2	Constructor & Destructor Documentation . . . . .	1249
6.204.2.1	~Comparable . . . . .	1249
6.204.3	Member Function Documentation . . . . .	1249
6.204.3.1	compareTo . . . . .	1249

6.204.3.2 equals . . . . .	1250
6.204.3.3 operator< . . . . .	1250
6.204.3.4 operator== . . . . .	1250
6.205decaf::util::Comparator< T > Class Template Reference . . . . .	1251
6.205.1 Detailed Description . . . . .	1251
6.205.2 Constructor & Destructor Documentation . . . . .	1252
6.205.2.1 ~Comparator . . . . .	1252
6.205.3 Member Function Documentation . . . . .	1252
6.205.3.1 compare . . . . .	1252
6.205.3.2 operator() . . . . .	1252
6.206activemq::util::CompositeData Class Reference . . . . .	1253
6.206.1 Detailed Description . . . . .	1253
6.206.2 Constructor & Destructor Documentation . . . . .	1254
6.206.2.1 CompositeData . . . . .	1254
6.206.2.2 ~CompositeData . . . . .	1254
6.206.3 Member Function Documentation . . . . .	1254
6.206.3.1 getComponents . . . . .	1254
6.206.3.2 getComponents . . . . .	1254
6.206.3.3 getFragment . . . . .	1254
6.206.3.4 getHost . . . . .	1254
6.206.3.5 getParameters . . . . .	1254
6.206.3.6 getPath . . . . .	1254
6.206.3.7 getScheme . . . . .	1254
6.206.3.8 setComponents . . . . .	1254
6.206.3.9 setFragment . . . . .	1254
6.206.3.10setHost . . . . .	1254
6.206.3.11setParameters . . . . .	1254
6.206.3.12setPath . . . . .	1254
6.206.3.13setScheme . . . . .	1254
6.206.3.14toURI . . . . .	1254
6.207activemq::threads::CompositeTask Class Reference . . . . .	1255
6.207.1 Detailed Description . . . . .	1255
6.207.2 Constructor & Destructor Documentation . . . . .	1255
6.207.2.1 ~CompositeTask . . . . .	1255
6.207.3 Member Function Documentation . . . . .	1255
6.207.3.1 isPending . . . . .	1255
6.208activemq::threads::CompositeTaskRunner Class Reference . . . . .	1256
6.208.1 Detailed Description . . . . .	1257
6.208.2 Constructor & Destructor Documentation . . . . .	1257
6.208.2.1 CompositeTaskRunner . . . . .	1257
6.208.2.2 ~CompositeTaskRunner . . . . .	1257
6.208.3 Member Function Documentation . . . . .	1257
6.208.3.1 addTask . . . . .	1257
6.208.3.2 iterate . . . . .	1257
6.208.3.3 removeTask . . . . .	1258
6.208.3.4 run . . . . .	1258
6.208.3.5 shutdown . . . . .	1258
6.208.3.6 shutdown . . . . .	1258
6.208.3.7 wakeup . . . . .	1258
6.209activemq::transport::CompositeTransport Class Reference . . . . .	1259

6.209.1 Detailed Description . . . . .	1259
6.209.2 Constructor & Destructor Documentation . . . . .	1259
6.209.2.1 ~CompositeTransport . . . . .	1259
6.209.3 Member Function Documentation . . . . .	1259
6.209.3.1 addURI . . . . .	1259
6.209.3.2 removeURI . . . . .	1260
6.210decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR > Class	
Template Reference . . . . .	1260
6.210.1 Detailed Description . . . . .	1261
6.210.2 Constructor & Destructor Documentation . . . . .	1261
6.210.2.1 ~ConcurrentMap . . . . .	1261
6.210.3 Member Function Documentation . . . . .	1261
6.210.3.1 putIfAbsent . . . . .	1261
6.210.3.2 remove . . . . .	1262
6.210.3.3 replace . . . . .	1263
6.210.3.4 replace . . . . .	1264
6.211decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >	
Class Template Reference . . . . .	1265
6.211.1 Detailed Description . . . . .	1269
6.211.2 Constructor & Destructor Documentation . . . . .	1269
6.211.2.1 ConcurrentStlMap . . . . .	1269
6.211.2.2 ConcurrentStlMap . . . . .	1270
6.211.2.3 ConcurrentStlMap . . . . .	1270
6.211.2.4 ~ConcurrentStlMap . . . . .	1270
6.211.3 Member Function Documentation . . . . .	1270
6.211.3.1 clear . . . . .	1270
6.211.3.2 containsKey . . . . .	1271
6.211.3.3 containsValue . . . . .	1271
6.211.3.4 copy . . . . .	1271
6.211.3.5 copy . . . . .	1272
6.211.3.6 equals . . . . .	1272
6.211.3.7 equals . . . . .	1272
6.211.3.8 get . . . . .	1272
6.211.3.9 get . . . . .	1273
6.211.3.10isEmpty . . . . .	1273
6.211.3.11keySet . . . . .	1273
6.211.3.12lock . . . . .	1274
6.211.3.13notify . . . . .	1274
6.211.3.14notifyAll . . . . .	1275
6.211.3.15put . . . . .	1275
6.211.3.16putAll . . . . .	1276
6.211.3.17putAll . . . . .	1276
6.211.3.18putIfAbsent . . . . .	1276
6.211.3.19remove . . . . .	1277
6.211.3.20remove . . . . .	1278
6.211.3.21replace . . . . .	1278
6.211.3.22replace . . . . .	1279
6.211.3.23size . . . . .	1279
6.211.3.24tryLock . . . . .	1280
6.211.3.25unlock . . . . .	1280



6.211.3.26	values	1280
6.211.3.27	wait	1281
6.211.3.28	wait	1281
6.211.3.29	wait	1282
6.212	decaf::util::concurrent::locks::Condition Class Reference	1282
6.212.1	Detailed Description	1283
6.212.2	Constructor & Destructor Documentation	1285
6.212.2.1	~Condition	1285
6.212.3	Member Function Documentation	1285
6.212.3.1	await	1285
6.212.3.2	await	1286
6.212.3.3	awaitNanos	1287
6.212.3.4	awaitUninterruptibly	1288
6.212.3.5	awaitUntil	1289
6.212.3.6	signal	1289
6.212.3.7	signalAll	1289
6.213	decaf::util::concurrent::ConditionHandle Class Reference	1290
6.213.1	Constructor & Destructor Documentation	1291
6.213.1.1	ConditionHandle	1291
6.213.1.2	~ConditionHandle	1291
6.213.1.3	ConditionHandle	1291
6.213.1.4	~ConditionHandle	1291
6.213.2	Field Documentation	1291
6.213.2.1	condition	1291
6.213.2.2	criticalSection	1291
6.213.2.3	generation	1291
6.213.2.4	mutex	1291
6.213.2.5	numWaiting	1291
6.213.2.6	numWake	1291
6.213.2.7	semaphore	1291
6.214	decaf::internal::util::concurrent::ConditionImpl Class Reference	1291
6.214.1	Member Function Documentation	1292
6.214.1.1	create	1292
6.214.1.2	destroy	1292
6.214.1.3	notify	1292
6.214.1.4	notifyAll	1293
6.214.1.5	wait	1293
6.214.1.6	wait	1293
6.215	decaf::net::ConnectException Class Reference	1293
6.215.1	Constructor & Destructor Documentation	1294
6.215.1.1	ConnectException	1294
6.215.1.2	ConnectException	1294
6.215.1.3	ConnectException	1294
6.215.1.4	ConnectException	1295
6.215.1.5	ConnectException	1295
6.215.1.6	ConnectException	1295
6.215.1.7	~ConnectException	1296
6.215.2	Member Function Documentation	1296
6.215.2.1	clone	1296
6.216	cms::Connection Class Reference	1296

6.216.1 Detailed Description . . . . .	1297
6.216.2 Constructor & Destructor Documentation . . . . .	1298
6.216.2.1 ~Connection . . . . .	1298
6.216.3 Member Function Documentation . . . . .	1298
6.216.3.1 close . . . . .	1298
6.216.3.2 createSession . . . . .	1298
6.216.3.3 createSession . . . . .	1299
6.216.3.4 getClientID . . . . .	1299
6.216.3.5 getExceptionListener . . . . .	1299
6.216.3.6 getMetaData . . . . .	1299
6.216.3.7 setClientID . . . . .	1300
6.216.3.8 setExceptionListener . . . . .	1300
6.217activemq::commands::ConnectionControl Class Reference . . . . .	1301
6.217.1 Constructor & Destructor Documentation . . . . .	1303
6.217.1.1 ConnectionControl . . . . .	1303
6.217.1.2 ~ConnectionControl . . . . .	1303
6.217.2 Member Function Documentation . . . . .	1303
6.217.2.1 cloneDataStructure . . . . .	1303
6.217.2.2 copyDataStructure . . . . .	1303
6.217.2.3 equals . . . . .	1303
6.217.2.4 getConnectedBrokers . . . . .	1304
6.217.2.5 getConnectedBrokers . . . . .	1304
6.217.2.6 getDataStructureType . . . . .	1304
6.217.2.7 getReconnectTo . . . . .	1305
6.217.2.8 getReconnectTo . . . . .	1305
6.217.2.9 isClose . . . . .	1305
6.217.2.10isExit . . . . .	1305
6.217.2.11isFaultTolerant . . . . .	1305
6.217.2.12isRebalanceConnection . . . . .	1305
6.217.2.13isResume . . . . .	1305
6.217.2.14isSuspend . . . . .	1305
6.217.2.15setClose . . . . .	1305
6.217.2.16setConnectedBrokers . . . . .	1305
6.217.2.17setExit . . . . .	1305
6.217.2.18setFaultTolerant . . . . .	1305
6.217.2.19setRebalanceConnection . . . . .	1305
6.217.2.20setReconnectTo . . . . .	1305
6.217.2.21setResume . . . . .	1305
6.217.2.22setSuspend . . . . .	1305
6.217.2.23toString . . . . .	1305
6.217.2.24visit . . . . .	1306
6.217.3 Field Documentation . . . . .	1306
6.217.3.1 close . . . . .	1306
6.217.3.2 connectedBrokers . . . . .	1306
6.217.3.3 exit . . . . .	1306
6.217.3.4 faultTolerant . . . . .	1306
6.217.3.5 ID_CONNECTIONCONTROL . . . . .	1306
6.217.3.6 rebalanceConnection . . . . .	1306
6.217.3.7 reconnectTo . . . . .	1306
6.217.3.8 resume . . . . .	1306

6.217.3.9 suspend . . . . .	1306
6.218activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	
Class Reference . . . . .	1307
6.218.1 Detailed Description . . . . .	1308
6.218.2 Constructor & Destructor Documentation . . . . .	1308
6.218.2.1 ConnectionControlMarshaller . . . . .	1308
6.218.2.2 ~ConnectionControlMarshaller . . . . .	1308
6.218.3 Member Function Documentation . . . . .	1308
6.218.3.1 createObject . . . . .	1308
6.218.3.2 getDataStructureType . . . . .	1308
6.218.3.3 looseMarshal . . . . .	1308
6.218.3.4 looseUnmarshal . . . . .	1309
6.218.3.5 tightMarshal1 . . . . .	1309
6.218.3.6 tightMarshal2 . . . . .	1310
6.218.3.7 tightUnmarshal . . . . .	1310
6.219activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	
Class Reference . . . . .	1311
6.219.1 Detailed Description . . . . .	1312
6.219.2 Constructor & Destructor Documentation . . . . .	1312
6.219.2.1 ConnectionControlMarshaller . . . . .	1312
6.219.2.2 ~ConnectionControlMarshaller . . . . .	1312
6.219.3 Member Function Documentation . . . . .	1312
6.219.3.1 createObject . . . . .	1312
6.219.3.2 getDataStructureType . . . . .	1312
6.219.3.3 looseMarshal . . . . .	1313
6.219.3.4 looseUnmarshal . . . . .	1313
6.219.3.5 tightMarshal1 . . . . .	1314
6.219.3.6 tightMarshal2 . . . . .	1314
6.219.3.7 tightUnmarshal . . . . .	1315
6.220activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller	
Class Reference . . . . .	1315
6.220.1 Detailed Description . . . . .	1316
6.220.2 Constructor & Destructor Documentation . . . . .	1316
6.220.2.1 ConnectionControlMarshaller . . . . .	1316
6.220.2.2 ~ConnectionControlMarshaller . . . . .	1316
6.220.3 Member Function Documentation . . . . .	1316
6.220.3.1 createObject . . . . .	1316
6.220.3.2 getDataStructureType . . . . .	1317
6.220.3.3 looseMarshal . . . . .	1317
6.220.3.4 looseUnmarshal . . . . .	1317
6.220.3.5 tightMarshal1 . . . . .	1318
6.220.3.6 tightMarshal2 . . . . .	1318
6.220.3.7 tightUnmarshal . . . . .	1319
6.221activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller	
Class Reference . . . . .	1319
6.221.1 Detailed Description . . . . .	1320
6.221.2 Constructor & Destructor Documentation . . . . .	1321
6.221.2.1 ConnectionControlMarshaller . . . . .	1321
6.221.2.2 ~ConnectionControlMarshaller . . . . .	1321
6.221.3 Member Function Documentation . . . . .	1321

6.221.3.1	createObject . . . . .	1321
6.221.3.2	getDataStructureType . . . . .	1321
6.221.3.3	looseMarshal . . . . .	1321
6.221.3.4	looseUnmarshal . . . . .	1322
6.221.3.5	tightMarshal1 . . . . .	1322
6.221.3.6	tightMarshal2 . . . . .	1323
6.221.3.7	tightUnmarshal . . . . .	1323
6.222	activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller	
	Class Reference . . . . .	1324
6.222.1	Detailed Description . . . . .	1325
6.222.2	Constructor & Destructor Documentation . . . . .	1325
6.222.2.1	ConnectionControlMarshaller . . . . .	1325
6.222.2.2	~ConnectionControlMarshaller . . . . .	1325
6.222.3	Member Function Documentation . . . . .	1325
6.222.3.1	createObject . . . . .	1325
6.222.3.2	getDataStructureType . . . . .	1325
6.222.3.3	looseMarshal . . . . .	1325
6.222.3.4	looseUnmarshal . . . . .	1326
6.222.3.5	tightMarshal1 . . . . .	1326
6.222.3.6	tightMarshal2 . . . . .	1327
6.222.3.7	tightUnmarshal . . . . .	1327
6.223	activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	
	Class Reference . . . . .	1328
6.223.1	Detailed Description . . . . .	1329
6.223.2	Constructor & Destructor Documentation . . . . .	1329
6.223.2.1	ConnectionControlMarshaller . . . . .	1329
6.223.2.2	~ConnectionControlMarshaller . . . . .	1329
6.223.3	Member Function Documentation . . . . .	1329
6.223.3.1	createObject . . . . .	1329
6.223.3.2	getDataStructureType . . . . .	1329
6.223.3.3	looseMarshal . . . . .	1330
6.223.3.4	looseUnmarshal . . . . .	1330
6.223.3.5	tightMarshal1 . . . . .	1331
6.223.3.6	tightMarshal2 . . . . .	1331
6.223.3.7	tightUnmarshal . . . . .	1332
6.224	activemq::commands::ConnectionError Class Reference . . . . .	1332
6.224.1	Constructor & Destructor Documentation . . . . .	1333
6.224.1.1	ConnectionError . . . . .	1333
6.224.1.2	~ConnectionError . . . . .	1333
6.224.2	Member Function Documentation . . . . .	1333
6.224.2.1	cloneDataStructure . . . . .	1333
6.224.2.2	copyDataStructure . . . . .	1334
6.224.2.3	equals . . . . .	1334
6.224.2.4	getConnectionId . . . . .	1334
6.224.2.5	getConnectionId . . . . .	1334
6.224.2.6	getDataStructureType . . . . .	1334
6.224.2.7	getException . . . . .	1335
6.224.2.8	getException . . . . .	1335
6.224.2.9	setConnectionId . . . . .	1335
6.224.2.10	setException . . . . .	1335

6.224.2.1 ltoString . . . . .	1335
6.224.2.12visit . . . . .	1335
6.224.3 Field Documentation . . . . .	1336
6.224.3.1 connectionId . . . . .	1336
6.224.3.2 exception . . . . .	1336
6.224.3.3 ID_CONNECTIONERROR . . . . .	1336
6.225activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	
Class Reference . . . . .	1336
6.225.1 Detailed Description . . . . .	1337
6.225.2 Constructor & Destructor Documentation . . . . .	1337
6.225.2.1 ConnectionErrorMarshaller . . . . .	1337
6.225.2.2 ~ConnectionErrorMarshaller . . . . .	1337
6.225.3 Member Function Documentation . . . . .	1337
6.225.3.1 createObject . . . . .	1337
6.225.3.2 getDataStructureType . . . . .	1337
6.225.3.3 looseMarshal . . . . .	1338
6.225.3.4 looseUnmarshal . . . . .	1338
6.225.3.5 tightMarshal1 . . . . .	1339
6.225.3.6 tightMarshal2 . . . . .	1339
6.225.3.7 tightUnmarshal . . . . .	1340
6.226activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	
Class Reference . . . . .	1340
6.226.1 Detailed Description . . . . .	1341
6.226.2 Constructor & Destructor Documentation . . . . .	1341
6.226.2.1 ConnectionErrorMarshaller . . . . .	1341
6.226.2.2 ~ConnectionErrorMarshaller . . . . .	1341
6.226.3 Member Function Documentation . . . . .	1341
6.226.3.1 createObject . . . . .	1341
6.226.3.2 getDataStructureType . . . . .	1342
6.226.3.3 looseMarshal . . . . .	1342
6.226.3.4 looseUnmarshal . . . . .	1342
6.226.3.5 tightMarshal1 . . . . .	1343
6.226.3.6 tightMarshal2 . . . . .	1343
6.226.3.7 tightUnmarshal . . . . .	1344
6.227activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	
Class Reference . . . . .	1344
6.227.1 Detailed Description . . . . .	1345
6.227.2 Constructor & Destructor Documentation . . . . .	1346
6.227.2.1 ConnectionErrorMarshaller . . . . .	1346
6.227.2.2 ~ConnectionErrorMarshaller . . . . .	1346
6.227.3 Member Function Documentation . . . . .	1346
6.227.3.1 createObject . . . . .	1346
6.227.3.2 getDataStructureType . . . . .	1346
6.227.3.3 looseMarshal . . . . .	1346
6.227.3.4 looseUnmarshal . . . . .	1347
6.227.3.5 tightMarshal1 . . . . .	1347
6.227.3.6 tightMarshal2 . . . . .	1348
6.227.3.7 tightUnmarshal . . . . .	1348
6.228activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	
Class Reference . . . . .	1349

6.228.1 Detailed Description . . . . .	1350
6.228.2 Constructor & Destructor Documentation . . . . .	1350
6.228.2.1 ConnectionErrorMarshaller . . . . .	1350
6.228.2.2 ~ConnectionErrorMarshaller . . . . .	1350
6.228.3 Member Function Documentation . . . . .	1350
6.228.3.1 createObject . . . . .	1350
6.228.3.2 getDataStructureType . . . . .	1350
6.228.3.3 looseMarshal . . . . .	1350
6.228.3.4 looseUnmarshal . . . . .	1351
6.228.3.5 tightMarshal1 . . . . .	1351
6.228.3.6 tightMarshal2 . . . . .	1352
6.228.3.7 tightUnmarshal . . . . .	1352
6.229activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller	
Class Reference . . . . .	1353
6.229.1 Detailed Description . . . . .	1354
6.229.2 Constructor & Destructor Documentation . . . . .	1354
6.229.2.1 ConnectionErrorMarshaller . . . . .	1354
6.229.2.2 ~ConnectionErrorMarshaller . . . . .	1354
6.229.3 Member Function Documentation . . . . .	1354
6.229.3.1 createObject . . . . .	1354
6.229.3.2 getDataStructureType . . . . .	1354
6.229.3.3 looseMarshal . . . . .	1355
6.229.3.4 looseUnmarshal . . . . .	1355
6.229.3.5 tightMarshal1 . . . . .	1356
6.229.3.6 tightMarshal2 . . . . .	1356
6.229.3.7 tightUnmarshal . . . . .	1357
6.230activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller	
Class Reference . . . . .	1357
6.230.1 Detailed Description . . . . .	1358
6.230.2 Constructor & Destructor Documentation . . . . .	1358
6.230.2.1 ConnectionErrorMarshaller . . . . .	1358
6.230.2.2 ~ConnectionErrorMarshaller . . . . .	1358
6.230.3 Member Function Documentation . . . . .	1358
6.230.3.1 createObject . . . . .	1358
6.230.3.2 getDataStructureType . . . . .	1359
6.230.3.3 looseMarshal . . . . .	1359
6.230.3.4 looseUnmarshal . . . . .	1359
6.230.3.5 tightMarshal1 . . . . .	1360
6.230.3.6 tightMarshal2 . . . . .	1360
6.230.3.7 tightUnmarshal . . . . .	1361
6.231cms::ConnectionFactory Class Reference . . . . .	1361
6.231.1 Detailed Description . . . . .	1362
6.231.2 Constructor & Destructor Documentation . . . . .	1362
6.231.2.1 ~ConnectionFactory . . . . .	1362
6.231.3 Member Function Documentation . . . . .	1362
6.231.3.1 createCMSConnectionFactory . . . . .	1362
6.231.3.2 createConnection . . . . .	1363
6.231.3.3 createConnection . . . . .	1364
6.231.3.4 createConnection . . . . .	1364
6.232activemq::commands::ConnectionId Class Reference . . . . .	1365

6.232.1 Member Typedef Documentation . . . . .	1366
6.232.1.1 COMPARATOR . . . . .	1366
6.232.2 Constructor & Destructor Documentation . . . . .	1366
6.232.2.1 ConnectionId . . . . .	1366
6.232.2.2 ConnectionId . . . . .	1366
6.232.2.3 ConnectionId . . . . .	1366
6.232.2.4 ConnectionId . . . . .	1366
6.232.2.5 ConnectionId . . . . .	1366
6.232.2.6 ~ConnectionId . . . . .	1366
6.232.3 Member Function Documentation . . . . .	1366
6.232.3.1 cloneDataStructure . . . . .	1366
6.232.3.2 compareTo . . . . .	1367
6.232.3.3 copyDataStructure . . . . .	1367
6.232.3.4 equals . . . . .	1367
6.232.3.5 equals . . . . .	1367
6.232.3.6 getDataStructureType . . . . .	1367
6.232.3.7 getValue . . . . .	1368
6.232.3.8 getValue . . . . .	1368
6.232.3.9 operator< . . . . .	1368
6.232.3.10operator= . . . . .	1368
6.232.3.11operator== . . . . .	1368
6.232.3.12setValue . . . . .	1368
6.232.3.13toString . . . . .	1368
6.232.4 Field Documentation . . . . .	1368
6.232.4.1 ID_CONNECTIONID . . . . .	1368
6.232.4.2 value . . . . .	1368
6.233activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	
Class Reference . . . . .	1368
6.233.1 Detailed Description . . . . .	1369
6.233.2 Constructor & Destructor Documentation . . . . .	1370
6.233.2.1 ConnectionIdMarshaller . . . . .	1370
6.233.2.2 ~ConnectionIdMarshaller . . . . .	1370
6.233.3 Member Function Documentation . . . . .	1370
6.233.3.1 createObject . . . . .	1370
6.233.3.2 getDataStructureType . . . . .	1370
6.233.3.3 looseMarshal . . . . .	1370
6.233.3.4 looseUnmarshal . . . . .	1371
6.233.3.5 tightMarshal1 . . . . .	1371
6.233.3.6 tightMarshal2 . . . . .	1372
6.233.3.7 tightUnmarshal . . . . .	1372
6.234activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	
Class Reference . . . . .	1373
6.234.1 Detailed Description . . . . .	1374
6.234.2 Constructor & Destructor Documentation . . . . .	1374
6.234.2.1 ConnectionIdMarshaller . . . . .	1374
6.234.2.2 ~ConnectionIdMarshaller . . . . .	1374
6.234.3 Member Function Documentation . . . . .	1374
6.234.3.1 createObject . . . . .	1374
6.234.3.2 getDataStructureType . . . . .	1374
6.234.3.3 looseMarshal . . . . .	1374

6.234.3.4	looseUnmarshal	1375
6.234.3.5	tightMarshal1	1375
6.234.3.6	tightMarshal2	1376
6.234.3.7	tightUnmarshal	1376
6.235	activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	
	Class Reference	1377
6.235.1	Detailed Description	1378
6.235.2	Constructor & Destructor Documentation	1378
6.235.2.1	ConnectionIdMarshaller	1378
6.235.2.2	~ConnectionIdMarshaller	1378
6.235.3	Member Function Documentation	1378
6.235.3.1	createObject	1378
6.235.3.2	getDataStructureType	1378
6.235.3.3	looseMarshal	1378
6.235.3.4	looseUnmarshal	1379
6.235.3.5	tightMarshal1	1379
6.235.3.6	tightMarshal2	1380
6.235.3.7	tightUnmarshal	1380
6.236	activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller	
	Class Reference	1381
6.236.1	Detailed Description	1382
6.236.2	Constructor & Destructor Documentation	1382
6.236.2.1	ConnectionIdMarshaller	1382
6.236.2.2	~ConnectionIdMarshaller	1382
6.236.3	Member Function Documentation	1382
6.236.3.1	createObject	1382
6.236.3.2	getDataStructureType	1382
6.236.3.3	looseMarshal	1382
6.236.3.4	looseUnmarshal	1383
6.236.3.5	tightMarshal1	1383
6.236.3.6	tightMarshal2	1384
6.236.3.7	tightUnmarshal	1384
6.237	activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller	
	Class Reference	1385
6.237.1	Detailed Description	1386
6.237.2	Constructor & Destructor Documentation	1386
6.237.2.1	ConnectionIdMarshaller	1386
6.237.2.2	~ConnectionIdMarshaller	1386
6.237.3	Member Function Documentation	1386
6.237.3.1	createObject	1386
6.237.3.2	getDataStructureType	1386
6.237.3.3	looseMarshal	1386
6.237.3.4	looseUnmarshal	1387
6.237.3.5	tightMarshal1	1387
6.237.3.6	tightMarshal2	1388
6.237.3.7	tightUnmarshal	1388
6.238	activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller	
	Class Reference	1389
6.238.1	Detailed Description	1390
6.238.2	Constructor & Destructor Documentation	1390



6.238.2.1	ConnectionIdMarshaller . . . . .	1390
6.238.2.2	~ConnectionIdMarshaller . . . . .	1390
6.238.3	Member Function Documentation . . . . .	1390
6.238.3.1	createObject . . . . .	1390
6.238.3.2	getDataStructureType . . . . .	1390
6.238.3.3	looseMarshal . . . . .	1390
6.238.3.4	looseUnmarshal . . . . .	1391
6.238.3.5	tightMarshal1 . . . . .	1391
6.238.3.6	tightMarshal2 . . . . .	1392
6.238.3.7	tightUnmarshal . . . . .	1392
6.239	activemq::commands::ConnectionInfo Class Reference . . . . .	1393
6.239.1	Constructor & Destructor Documentation . . . . .	1395
6.239.1.1	ConnectionInfo . . . . .	1395
6.239.1.2	~ConnectionInfo . . . . .	1395
6.239.2	Member Function Documentation . . . . .	1395
6.239.2.1	cloneDataStructure . . . . .	1395
6.239.2.2	copyDataStructure . . . . .	1395
6.239.2.3	createRemoveCommand . . . . .	1395
6.239.2.4	equals . . . . .	1395
6.239.2.5	getBrokerPath . . . . .	1396
6.239.2.6	getBrokerPath . . . . .	1396
6.239.2.7	getClientId . . . . .	1396
6.239.2.8	getClientId . . . . .	1396
6.239.2.9	getConnectionId . . . . .	1396
6.239.2.10	getConnectionId . . . . .	1396
6.239.2.11	getDataStructureType . . . . .	1396
6.239.2.12	getPassword . . . . .	1397
6.239.2.13	getPassword . . . . .	1397
6.239.2.14	getUserName . . . . .	1397
6.239.2.15	getUserName . . . . .	1397
6.239.2.16	isBrokerMasterConnector . . . . .	1397
6.239.2.17	isClientMaster . . . . .	1397
6.239.2.18	isConnectionInfo . . . . .	1397
6.239.2.19	isFaultTolerant . . . . .	1398
6.239.2.20	isManageable . . . . .	1398
6.239.2.21	setBrokerMasterConnector . . . . .	1398
6.239.2.22	setBrokerPath . . . . .	1398
6.239.2.23	setClientId . . . . .	1398
6.239.2.24	setClientMaster . . . . .	1398
6.239.2.25	setConnectionId . . . . .	1398
6.239.2.26	setFaultTolerant . . . . .	1398
6.239.2.27	setManageable . . . . .	1398
6.239.2.28	setPassword . . . . .	1398
6.239.2.29	setUserName . . . . .	1398
6.239.2.30	toString . . . . .	1398
6.239.2.31	visit . . . . .	1399
6.239.3	Field Documentation . . . . .	1399
6.239.3.1	brokerMasterConnector . . . . .	1399
6.239.3.2	brokerPath . . . . .	1399
6.239.3.3	clientId . . . . .	1399

6.239.3.4	clientMaster . . . . .	1399
6.239.3.5	connectionId . . . . .	1399
6.239.3.6	faultTolerant . . . . .	1399
6.239.3.7	ID_CONNECTIONINFO . . . . .	1399
6.239.3.8	manageable . . . . .	1399
6.239.3.9	password . . . . .	1399
6.239.3.10	userName . . . . .	1399
6.240	activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	
	Class Reference . . . . .	1400
6.240.1	Detailed Description . . . . .	1401
6.240.2	Constructor & Destructor Documentation . . . . .	1401
	6.240.2.1 ConnectionInfoMarshaller . . . . .	1401
	6.240.2.2 ~ConnectionInfoMarshaller . . . . .	1401
6.240.3	Member Function Documentation . . . . .	1401
	6.240.3.1 createObject . . . . .	1401
	6.240.3.2 getDataStructureType . . . . .	1401
	6.240.3.3 looseMarshal . . . . .	1401
	6.240.3.4 looseUnmarshal . . . . .	1402
	6.240.3.5 tightMarshal1 . . . . .	1402
	6.240.3.6 tightMarshal2 . . . . .	1403
	6.240.3.7 tightUnmarshal . . . . .	1403
6.241	activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	
	Class Reference . . . . .	1404
6.241.1	Detailed Description . . . . .	1405
6.241.2	Constructor & Destructor Documentation . . . . .	1405
	6.241.2.1 ConnectionInfoMarshaller . . . . .	1405
	6.241.2.2 ~ConnectionInfoMarshaller . . . . .	1405
6.241.3	Member Function Documentation . . . . .	1405
	6.241.3.1 createObject . . . . .	1405
	6.241.3.2 getDataStructureType . . . . .	1405
	6.241.3.3 looseMarshal . . . . .	1406
	6.241.3.4 looseUnmarshal . . . . .	1406
	6.241.3.5 tightMarshal1 . . . . .	1407
	6.241.3.6 tightMarshal2 . . . . .	1407
	6.241.3.7 tightUnmarshal . . . . .	1408
6.242	activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	
	Class Reference . . . . .	1408
6.242.1	Detailed Description . . . . .	1409
6.242.2	Constructor & Destructor Documentation . . . . .	1409
	6.242.2.1 ConnectionInfoMarshaller . . . . .	1409
	6.242.2.2 ~ConnectionInfoMarshaller . . . . .	1409
6.242.3	Member Function Documentation . . . . .	1409
	6.242.3.1 createObject . . . . .	1409
	6.242.3.2 getDataStructureType . . . . .	1410
	6.242.3.3 looseMarshal . . . . .	1410
	6.242.3.4 looseUnmarshal . . . . .	1410
	6.242.3.5 tightMarshal1 . . . . .	1411
	6.242.3.6 tightMarshal2 . . . . .	1411
	6.242.3.7 tightUnmarshal . . . . .	1412

6.243activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller	
Class Reference . . . . .	1412
6.243.1 Detailed Description . . . . .	1413
6.243.2 Constructor & Destructor Documentation . . . . .	1414
6.243.2.1 ConnectionInfoMarshaller . . . . .	1414
6.243.2.2 ~ConnectionInfoMarshaller . . . . .	1414
6.243.3 Member Function Documentation . . . . .	1414
6.243.3.1 createObject . . . . .	1414
6.243.3.2 getDataStructureType . . . . .	1414
6.243.3.3 looseMarshal . . . . .	1414
6.243.3.4 looseUnmarshal . . . . .	1415
6.243.3.5 tightMarshal1 . . . . .	1415
6.243.3.6 tightMarshal2 . . . . .	1416
6.243.3.7 tightUnmarshal . . . . .	1416
6.244activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller	
Class Reference . . . . .	1417
6.244.1 Detailed Description . . . . .	1418
6.244.2 Constructor & Destructor Documentation . . . . .	1418
6.244.2.1 ConnectionInfoMarshaller . . . . .	1418
6.244.2.2 ~ConnectionInfoMarshaller . . . . .	1418
6.244.3 Member Function Documentation . . . . .	1418
6.244.3.1 createObject . . . . .	1418
6.244.3.2 getDataStructureType . . . . .	1418
6.244.3.3 looseMarshal . . . . .	1418
6.244.3.4 looseUnmarshal . . . . .	1419
6.244.3.5 tightMarshal1 . . . . .	1419
6.244.3.6 tightMarshal2 . . . . .	1420
6.244.3.7 tightUnmarshal . . . . .	1420
6.245activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller	
Class Reference . . . . .	1421
6.245.1 Detailed Description . . . . .	1422
6.245.2 Constructor & Destructor Documentation . . . . .	1422
6.245.2.1 ConnectionInfoMarshaller . . . . .	1422
6.245.2.2 ~ConnectionInfoMarshaller . . . . .	1422
6.245.3 Member Function Documentation . . . . .	1422
6.245.3.1 createObject . . . . .	1422
6.245.3.2 getDataStructureType . . . . .	1422
6.245.3.3 looseMarshal . . . . .	1423
6.245.3.4 looseUnmarshal . . . . .	1423
6.245.3.5 tightMarshal1 . . . . .	1424
6.245.3.6 tightMarshal2 . . . . .	1424
6.245.3.7 tightUnmarshal . . . . .	1425
6.246cms::ConnectionMetaData Class Reference . . . . .	1425
6.246.1 Detailed Description . . . . .	1426
6.246.2 Constructor & Destructor Documentation . . . . .	1426
6.246.2.1 ~ConnectionMetaData . . . . .	1426
6.246.3 Member Function Documentation . . . . .	1426
6.246.3.1 getCMSMajorVersion . . . . .	1426
6.246.3.2 getCMSMinorVersion . . . . .	1427
6.246.3.3 getCMSProviderName . . . . .	1427

6.246.3.4	getCMSVersion	1427
6.246.3.5	getCMSXPropertyNames	1428
6.246.3.6	getProviderMajorVersion	1428
6.246.3.7	getProviderMinorVersion	1428
6.246.3.8	getProviderVersion	1429
6.247	activemq::state::ConnectionState Class Reference	1429
6.247.1	Constructor & Destructor Documentation	1431
6.247.1.1	ConnectionState	1431
6.247.1.2	~ConnectionState	1431
6.247.2	Member Function Documentation	1431
6.247.2.1	addSession	1431
6.247.2.2	addTempDestination	1431
6.247.2.3	addTransactionState	1431
6.247.2.4	checkShutdown	1431
6.247.2.5	getInfo	1431
6.247.2.6	getRecoveringPullConsumers	1431
6.247.2.7	getSessionState	1431
6.247.2.8	getSessionStates	1431
6.247.2.9	getTempDesinations	1431
6.247.2.10	getTransactionState	1431
6.247.2.11	getTransactionStates	1432
6.247.2.12	isConnectionInterruptProcessingComplete	1432
6.247.2.13	removeSession	1432
6.247.2.14	removeTempDestination	1432
6.247.2.15	removeTransactionState	1432
6.247.2.16	reset	1432
6.247.2.17	setConnectionInterruptProcessingComplete	1432
6.247.2.18	shutdown	1432
6.247.2.19	toString	1432
6.248	activemq::state::ConnectionStateTracker Class Reference	1432
6.248.1	Constructor & Destructor Documentation	1435
6.248.1.1	ConnectionStateTracker	1435
6.248.1.2	~ConnectionStateTracker	1435
6.248.2	Member Function Documentation	1435
6.248.2.1	connectionInterruptProcessingComplete	1435
6.248.2.2	getMaxCacheSize	1435
6.248.2.3	isRestoreConsumers	1435
6.248.2.4	isRestoreProducers	1435
6.248.2.5	isRestoreSessions	1435
6.248.2.6	isRestoreTransaction	1435
6.248.2.7	isTrackMessages	1435
6.248.2.8	isTrackTransactionProducers	1435
6.248.2.9	isTrackTransactions	1435
6.248.2.10	processBeginTransaction	1435
6.248.2.11	processCommitTransactionOnePhase	1436
6.248.2.12	processCommitTransactionTwoPhase	1436
6.248.2.13	processConnectionInfo	1436
6.248.2.14	processConsumerInfo	1436
6.248.2.15	processDestinationInfo	1436
6.248.2.16	processEndTransaction	1436

6.248.2.17	processMessage	1437
6.248.2.18	processMessageAck	1437
6.248.2.19	processPrepareTransaction	1437
6.248.2.20	processProducerInfo	1437
6.248.2.21	processRemoveConnection	1437
6.248.2.22	processRemoveConsumer	1437
6.248.2.23	processRemoveDestination	1438
6.248.2.24	processRemoveProducer	1438
6.248.2.25	processRemoveSession	1438
6.248.2.26	processRollbackTransaction	1438
6.248.2.27	processSessionInfo	1438
6.248.2.28	restore	1439
6.248.2.29	setMaxCacheSize	1439
6.248.2.30	setRestoreConsumers	1439
6.248.2.31	setRestoreProducers	1439
6.248.2.32	setRestoreSessions	1439
6.248.2.33	setRestoreTransaction	1439
6.248.2.34	setTrackMessages	1439
6.248.2.35	setTrackTransactionProducers	1439
6.248.2.36	setTrackTransactions	1439
6.248.2.37	track	1439
6.248.2.38	trackBack	1439
6.248.2.39	transportInterrupted	1439
6.248.3	Friends And Related Function Documentation	1439
6.248.3.1	RemoveTransactionAction	1439
6.249	decaf::util::logging::ConsoleHandler Class Reference	1439
6.249.1	Detailed Description	1440
6.249.2	Constructor & Destructor Documentation	1440
6.249.2.1	ConsoleHandler	1440
6.249.2.2	~ConsoleHandler	1440
6.249.3	Member Function Documentation	1440
6.249.3.1	close	1440
6.249.3.2	publish	1441
6.250	activemq::commands::ConsumerControl Class Reference	1441
6.250.1	Constructor & Destructor Documentation	1443
6.250.1.1	ConsumerControl	1443
6.250.1.2	~ConsumerControl	1443
6.250.2	Member Function Documentation	1443
6.250.2.1	cloneDataStructure	1443
6.250.2.2	copyDataStructure	1443
6.250.2.3	equals	1443
6.250.2.4	getConsumerId	1444
6.250.2.5	getConsumerId	1444
6.250.2.6	getDataStructureType	1444
6.250.2.7	getDestination	1445
6.250.2.8	getDestination	1445
6.250.2.9	getPrefetch	1445
6.250.2.10	isClose	1445
6.250.2.11	isFlush	1445
6.250.2.12	isStart	1445

6.250.2.13	isStop	1445
6.250.2.14	setClose	1445
6.250.2.15	setConsumerId	1445
6.250.2.16	setDestination	1445
6.250.2.17	setFlush	1445
6.250.2.18	setPrefetch	1445
6.250.2.19	setStart	1445
6.250.2.20	setStop	1445
6.250.2.21	toString	1445
6.250.2.22	visit	1446
6.250.3	Field Documentation	1446
6.250.3.1	close	1446
6.250.3.2	consumerId	1446
6.250.3.3	destination	1446
6.250.3.4	flush	1446
6.250.3.5	ID_CONSUMERCONTROL	1446
6.250.3.6	prefetch	1446
6.250.3.7	start	1446
6.250.3.8	stop	1446
6.251	activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller	
	Class Reference	1447
6.251.1	Detailed Description	1448
6.251.2	Constructor & Destructor Documentation	1448
6.251.2.1	ConsumerControlMarshaller	1448
6.251.2.2	~ConsumerControlMarshaller	1448
6.251.3	Member Function Documentation	1448
6.251.3.1	createObject	1448
6.251.3.2	getDataStructureType	1448
6.251.3.3	looseMarshal	1448
6.251.3.4	looseUnmarshal	1449
6.251.3.5	tightMarshal1	1449
6.251.3.6	tightMarshal2	1450
6.251.3.7	tightUnmarshal	1450
6.252	activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller	
	Class Reference	1451
6.252.1	Detailed Description	1452
6.252.2	Constructor & Destructor Documentation	1452
6.252.2.1	ConsumerControlMarshaller	1452
6.252.2.2	~ConsumerControlMarshaller	1452
6.252.3	Member Function Documentation	1452
6.252.3.1	createObject	1452
6.252.3.2	getDataStructureType	1452
6.252.3.3	looseMarshal	1453
6.252.3.4	looseUnmarshal	1453
6.252.3.5	tightMarshal1	1454
6.252.3.6	tightMarshal2	1454
6.252.3.7	tightUnmarshal	1455
6.253	activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller	
	Class Reference	1455
6.253.1	Detailed Description	1456

6.253.2 Constructor & Destructor Documentation . . . . .	1456
6.253.2.1 ConsumerControlMarshaller . . . . .	1456
6.253.2.2 ~ConsumerControlMarshaller . . . . .	1456
6.253.3 Member Function Documentation . . . . .	1456
6.253.3.1 createObject . . . . .	1456
6.253.3.2 getDataStructureType . . . . .	1457
6.253.3.3 looseMarshal . . . . .	1457
6.253.3.4 looseUnmarshal . . . . .	1457
6.253.3.5 tightMarshal1 . . . . .	1458
6.253.3.6 tightMarshal2 . . . . .	1458
6.253.3.7 tightUnmarshal . . . . .	1459
6.254activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller	
Class Reference . . . . .	1459
6.254.1 Detailed Description . . . . .	1460
6.254.2 Constructor & Destructor Documentation . . . . .	1461
6.254.2.1 ConsumerControlMarshaller . . . . .	1461
6.254.2.2 ~ConsumerControlMarshaller . . . . .	1461
6.254.3 Member Function Documentation . . . . .	1461
6.254.3.1 createObject . . . . .	1461
6.254.3.2 getDataStructureType . . . . .	1461
6.254.3.3 looseMarshal . . . . .	1461
6.254.3.4 looseUnmarshal . . . . .	1462
6.254.3.5 tightMarshal1 . . . . .	1462
6.254.3.6 tightMarshal2 . . . . .	1463
6.254.3.7 tightUnmarshal . . . . .	1463
6.255activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller	
Class Reference . . . . .	1464
6.255.1 Detailed Description . . . . .	1465
6.255.2 Constructor & Destructor Documentation . . . . .	1465
6.255.2.1 ConsumerControlMarshaller . . . . .	1465
6.255.2.2 ~ConsumerControlMarshaller . . . . .	1465
6.255.3 Member Function Documentation . . . . .	1465
6.255.3.1 createObject . . . . .	1465
6.255.3.2 getDataStructureType . . . . .	1465
6.255.3.3 looseMarshal . . . . .	1465
6.255.3.4 looseUnmarshal . . . . .	1466
6.255.3.5 tightMarshal1 . . . . .	1466
6.255.3.6 tightMarshal2 . . . . .	1467
6.255.3.7 tightUnmarshal . . . . .	1467
6.256activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller	
Class Reference . . . . .	1468
6.256.1 Detailed Description . . . . .	1469
6.256.2 Constructor & Destructor Documentation . . . . .	1469
6.256.2.1 ConsumerControlMarshaller . . . . .	1469
6.256.2.2 ~ConsumerControlMarshaller . . . . .	1469
6.256.3 Member Function Documentation . . . . .	1469
6.256.3.1 createObject . . . . .	1469
6.256.3.2 getDataStructureType . . . . .	1469
6.256.3.3 looseMarshal . . . . .	1470
6.256.3.4 looseUnmarshal . . . . .	1470

6.256.3.5	tightMarshal1	1471
6.256.3.6	tightMarshal2	1471
6.256.3.7	tightUnmarshal	1472
6.257	activemq::commands::ConsumerId Class Reference	1472
6.257.1	Member Typedef Documentation	1474
6.257.1.1	COMPARATOR	1474
6.257.2	Constructor & Destructor Documentation	1474
6.257.2.1	ConsumerId	1474
6.257.2.2	ConsumerId	1474
6.257.2.3	ConsumerId	1474
6.257.2.4	~ConsumerId	1474
6.257.3	Member Function Documentation	1474
6.257.3.1	cloneDataStructure	1474
6.257.3.2	compareTo	1474
6.257.3.3	copyDataStructure	1474
6.257.3.4	equals	1475
6.257.3.5	equals	1475
6.257.3.6	getConnectionId	1475
6.257.3.7	getConnectionId	1475
6.257.3.8	getDataStructureType	1475
6.257.3.9	getParentId	1476
6.257.3.10	getSessionId	1476
6.257.3.11	getValue	1476
6.257.3.12	operator<	1476
6.257.3.13	operator=	1476
6.257.3.14	operator==	1476
6.257.3.15	setConnectionId	1476
6.257.3.16	setSessionId	1476
6.257.3.17	setValue	1476
6.257.3.18	toString	1476
6.257.4	Field Documentation	1476
6.257.4.1	connectionId	1476
6.257.4.2	ID_CONSUMERID	1476
6.257.4.3	sessionId	1477
6.257.4.4	value	1477
6.258	activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller Class Reference	1477
6.258.1	Detailed Description	1478
6.258.2	Constructor & Destructor Documentation	1478
6.258.2.1	ConsumerIdMarshaller	1478
6.258.2.2	~ConsumerIdMarshaller	1478
6.258.3	Member Function Documentation	1478
6.258.3.1	createObject	1478
6.258.3.2	getDataStructureType	1478
6.258.3.3	looseMarshal	1479
6.258.3.4	looseUnmarshal	1479
6.258.3.5	tightMarshal1	1479
6.258.3.6	tightMarshal2	1480
6.258.3.7	tightUnmarshal	1480



6.259	activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller Class	
	Reference	1481
6.259.1	Detailed Description	1482
6.259.2	Constructor & Destructor Documentation	1482
6.259.2.1	ConsumerIdMarshaller	1482
6.259.2.2	~ConsumerIdMarshaller	1482
6.259.3	Member Function Documentation	1482
6.259.3.1	createObject	1482
6.259.3.2	getDataStructureType	1482
6.259.3.3	looseMarshal	1483
6.259.3.4	looseUnmarshal	1483
6.259.3.5	tightMarshal1	1484
6.259.3.6	tightMarshal2	1484
6.259.3.7	tightUnmarshal	1485
6.260	activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller Class	
	Reference	1485
6.260.1	Detailed Description	1486
6.260.2	Constructor & Destructor Documentation	1486
6.260.2.1	ConsumerIdMarshaller	1486
6.260.2.2	~ConsumerIdMarshaller	1486
6.260.3	Member Function Documentation	1486
6.260.3.1	createObject	1486
6.260.3.2	getDataStructureType	1487
6.260.3.3	looseMarshal	1487
6.260.3.4	looseUnmarshal	1487
6.260.3.5	tightMarshal1	1488
6.260.3.6	tightMarshal2	1488
6.260.3.7	tightUnmarshal	1489
6.261	activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller Class	
	Reference	1489
6.261.1	Detailed Description	1490
6.261.2	Constructor & Destructor Documentation	1490
6.261.2.1	ConsumerIdMarshaller	1490
6.261.2.2	~ConsumerIdMarshaller	1490
6.261.3	Member Function Documentation	1490
6.261.3.1	createObject	1490
6.261.3.2	getDataStructureType	1491
6.261.3.3	looseMarshal	1491
6.261.3.4	looseUnmarshal	1491
6.261.3.5	tightMarshal1	1492
6.261.3.6	tightMarshal2	1492
6.261.3.7	tightUnmarshal	1493
6.262	activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller Class	
	Reference	1493
6.262.1	Detailed Description	1494
6.262.2	Constructor & Destructor Documentation	1494
6.262.2.1	ConsumerIdMarshaller	1494
6.262.2.2	~ConsumerIdMarshaller	1494
6.262.3	Member Function Documentation	1494
6.262.3.1	createObject	1494

6.262.3.2	getDataStructureType . . . . .	1495
6.262.3.3	looseMarshal . . . . .	1495
6.262.3.4	looseUnmarshal . . . . .	1495
6.262.3.5	tightMarshal1 . . . . .	1496
6.262.3.6	tightMarshal2 . . . . .	1496
6.262.3.7	tightUnmarshal . . . . .	1497
6.263	activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller Class	
	Reference . . . . .	1497
6.263.1	Detailed Description . . . . .	1498
6.263.2	Constructor & Destructor Documentation . . . . .	1498
6.263.2.1	ConsumerIdMarshaller . . . . .	1498
6.263.2.2	~ConsumerIdMarshaller . . . . .	1498
6.263.3	Member Function Documentation . . . . .	1498
6.263.3.1	createObject . . . . .	1498
6.263.3.2	getDataStructureType . . . . .	1499
6.263.3.3	looseMarshal . . . . .	1499
6.263.3.4	looseUnmarshal . . . . .	1499
6.263.3.5	tightMarshal1 . . . . .	1500
6.263.3.6	tightMarshal2 . . . . .	1500
6.263.3.7	tightUnmarshal . . . . .	1501
6.264	activemq::commands::ConsumerInfo Class Reference . . . . .	1501
6.264.1	Constructor & Destructor Documentation . . . . .	1504
6.264.1.1	ConsumerInfo . . . . .	1504
6.264.1.2	~ConsumerInfo . . . . .	1504
6.264.2	Member Function Documentation . . . . .	1504
6.264.2.1	cloneDataStructure . . . . .	1504
6.264.2.2	copyDataStructure . . . . .	1504
6.264.2.3	createRemoveCommand . . . . .	1504
6.264.2.4	equals . . . . .	1504
6.264.2.5	getAdditionalPredicate . . . . .	1505
6.264.2.6	getAdditionalPredicate . . . . .	1505
6.264.2.7	getBrokerPath . . . . .	1505
6.264.2.8	getBrokerPath . . . . .	1505
6.264.2.9	getConsumerId . . . . .	1505
6.264.2.10	getConsumerId . . . . .	1505
6.264.2.11	getDataStructureType . . . . .	1505
6.264.2.12	getDestination . . . . .	1506
6.264.2.13	getDestination . . . . .	1506
6.264.2.14	getMaximumPendingMessageLimit . . . . .	1506
6.264.2.15	getNetworkConsumerPath . . . . .	1506
6.264.2.16	getNetworkConsumerPath . . . . .	1506
6.264.2.17	getPrefetchSize . . . . .	1506
6.264.2.18	getPriority . . . . .	1506
6.264.2.19	getSelector . . . . .	1506
6.264.2.20	getSelector . . . . .	1506
6.264.2.21	getSubscriptionName . . . . .	1506
6.264.2.22	getSubscriptionName . . . . .	1506
6.264.2.23	isBrowser . . . . .	1506
6.264.2.24	isConsumerInfo . . . . .	1506
6.264.2.25	isDispatchAsync . . . . .	1507

6.264.2.26	isExclusive	1507
6.264.2.27	isNetworkSubscription	1507
6.264.2.28	isNoLocal	1507
6.264.2.29	isNoRangeAcks	1507
6.264.2.30	isOptimizedAcknowledge	1507
6.264.2.31	isRetroactive	1507
6.264.2.32	setAdditionalPredicate	1507
6.264.2.33	setBrokerPath	1507
6.264.2.34	setBrowser	1507
6.264.2.35	setConsumerId	1507
6.264.2.36	setDestination	1507
6.264.2.37	setDispatchAsync	1507
6.264.2.38	setExclusive	1507
6.264.2.39	setMaximumPendingMessageLimit	1507
6.264.2.40	setNetworkConsumerPath	1507
6.264.2.41	setNetworkSubscription	1507
6.264.2.42	setNoLocal	1507
6.264.2.43	setNoRangeAcks	1507
6.264.2.44	setOptimizedAcknowledge	1507
6.264.2.45	setPrefetchSize	1507
6.264.2.46	setPriority	1507
6.264.2.47	setRetroactive	1507
6.264.2.48	setSelector	1507
6.264.2.49	setSubscriptionName	1507
6.264.2.50	toString	1507
6.264.2.51	visit	1508
6.264.3	Field Documentation	1509
6.264.3.1	additionalPredicate	1509
6.264.3.2	brokerPath	1509
6.264.3.3	browser	1509
6.264.3.4	consumerId	1509
6.264.3.5	destination	1509
6.264.3.6	dispatchAsync	1509
6.264.3.7	exclusive	1509
6.264.3.8	ID_CONSUMERINFO	1509
6.264.3.9	maximumPendingMessageLimit	1509
6.264.3.10	networkConsumerPath	1509
6.264.3.11	networkSubscription	1509
6.264.3.12	noLocal	1509
6.264.3.13	noRangeAcks	1509
6.264.3.14	optimizedAcknowledge	1509
6.264.3.15	prefetchSize	1509
6.264.3.16	priority	1509
6.264.3.17	retroactive	1509
6.264.3.18	selector	1509
6.264.3.19	subscriptionName	1509
6.265	activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	
	Class Reference	1510
6.265.1	Detailed Description	1511
6.265.2	Constructor & Destructor Documentation	1511

6.265.2.1 ConsumerInfoMarshaller . . . . .	1511
6.265.2.2 ~ConsumerInfoMarshaller . . . . .	1511
6.265.3 Member Function Documentation . . . . .	1511
6.265.3.1 createObject . . . . .	1511
6.265.3.2 getDataStructureType . . . . .	1511
6.265.3.3 looseMarshal . . . . .	1511
6.265.3.4 looseUnmarshal . . . . .	1512
6.265.3.5 tightMarshal1 . . . . .	1512
6.265.3.6 tightMarshal2 . . . . .	1513
6.265.3.7 tightUnmarshal . . . . .	1513
6.266activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	
Class Reference . . . . .	1514
6.266.1 Detailed Description . . . . .	1515
6.266.2 Constructor & Destructor Documentation . . . . .	1515
6.266.2.1 ConsumerInfoMarshaller . . . . .	1515
6.266.2.2 ~ConsumerInfoMarshaller . . . . .	1515
6.266.3 Member Function Documentation . . . . .	1515
6.266.3.1 createObject . . . . .	1515
6.266.3.2 getDataStructureType . . . . .	1515
6.266.3.3 looseMarshal . . . . .	1516
6.266.3.4 looseUnmarshal . . . . .	1516
6.266.3.5 tightMarshal1 . . . . .	1517
6.266.3.6 tightMarshal2 . . . . .	1517
6.266.3.7 tightUnmarshal . . . . .	1518
6.267activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	
Class Reference . . . . .	1518
6.267.1 Detailed Description . . . . .	1519
6.267.2 Constructor & Destructor Documentation . . . . .	1519
6.267.2.1 ConsumerInfoMarshaller . . . . .	1519
6.267.2.2 ~ConsumerInfoMarshaller . . . . .	1519
6.267.3 Member Function Documentation . . . . .	1519
6.267.3.1 createObject . . . . .	1519
6.267.3.2 getDataStructureType . . . . .	1520
6.267.3.3 looseMarshal . . . . .	1520
6.267.3.4 looseUnmarshal . . . . .	1520
6.267.3.5 tightMarshal1 . . . . .	1521
6.267.3.6 tightMarshal2 . . . . .	1521
6.267.3.7 tightUnmarshal . . . . .	1522
6.268activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	
Class Reference . . . . .	1522
6.268.1 Detailed Description . . . . .	1523
6.268.2 Constructor & Destructor Documentation . . . . .	1524
6.268.2.1 ConsumerInfoMarshaller . . . . .	1524
6.268.2.2 ~ConsumerInfoMarshaller . . . . .	1524
6.268.3 Member Function Documentation . . . . .	1524
6.268.3.1 createObject . . . . .	1524
6.268.3.2 getDataStructureType . . . . .	1524
6.268.3.3 looseMarshal . . . . .	1524
6.268.3.4 looseUnmarshal . . . . .	1525
6.268.3.5 tightMarshal1 . . . . .	1525

6.268.3.6	tightMarshal2	1526
6.268.3.7	tightUnmarshal	1526
6.269	activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller	
	Class Reference	1527
6.269.1	Detailed Description	1528
6.269.2	Constructor & Destructor Documentation	1528
6.269.2.1	ConsumerInfoMarshaller	1528
6.269.2.2	~ConsumerInfoMarshaller	1528
6.269.3	Member Function Documentation	1528
6.269.3.1	createObject	1528
6.269.3.2	getDataStructureType	1528
6.269.3.3	looseMarshal	1528
6.269.3.4	looseUnmarshal	1529
6.269.3.5	tightMarshal1	1529
6.269.3.6	tightMarshal2	1530
6.269.3.7	tightUnmarshal	1530
6.270	activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller	
	Class Reference	1531
6.270.1	Detailed Description	1532
6.270.2	Constructor & Destructor Documentation	1532
6.270.2.1	ConsumerInfoMarshaller	1532
6.270.2.2	~ConsumerInfoMarshaller	1532
6.270.3	Member Function Documentation	1532
6.270.3.1	createObject	1532
6.270.3.2	getDataStructureType	1532
6.270.3.3	looseMarshal	1533
6.270.3.4	looseUnmarshal	1533
6.270.3.5	tightMarshal1	1534
6.270.3.6	tightMarshal2	1534
6.270.3.7	tightUnmarshal	1535
6.271	activemq::state::ConsumerState Class Reference	1535
6.271.1	Constructor & Destructor Documentation	1536
6.271.1.1	ConsumerState	1536
6.271.1.2	~ConsumerState	1536
6.271.2	Member Function Documentation	1536
6.271.2.1	getInfo	1536
6.271.2.2	toString	1536
6.272	activemq::commands::ControlCommand Class Reference	1536
6.272.1	Constructor & Destructor Documentation	1537
6.272.1.1	ControlCommand	1537
6.272.1.2	~ControlCommand	1537
6.272.2	Member Function Documentation	1537
6.272.2.1	cloneDataStructure	1537
6.272.2.2	copyDataStructure	1537
6.272.2.3	equals	1538
6.272.2.4	getCommand	1538
6.272.2.5	getCommand	1538
6.272.2.6	getDataStructureType	1538
6.272.2.7	setCommand	1538
6.272.2.8	toString	1538

6.272.2.9 visit . . . . .	1539
6.272.3 Field Documentation . . . . .	1539
6.272.3.1 command . . . . .	1539
6.272.3.2 ID_CONTROLCOMMAND . . . . .	1539
6.273activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller	
Class Reference . . . . .	1539
6.273.1 Detailed Description . . . . .	1540
6.273.2 Constructor & Destructor Documentation . . . . .	1541
6.273.2.1 ControlCommandMarshaller . . . . .	1541
6.273.2.2 ~ControlCommandMarshaller . . . . .	1541
6.273.3 Member Function Documentation . . . . .	1541
6.273.3.1 createObject . . . . .	1541
6.273.3.2 getDataStructureType . . . . .	1541
6.273.3.3 looseMarshal . . . . .	1541
6.273.3.4 looseUnmarshal . . . . .	1542
6.273.3.5 tightMarshal1 . . . . .	1542
6.273.3.6 tightMarshal2 . . . . .	1543
6.273.3.7 tightUnmarshal . . . . .	1543
6.274activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller	
Class Reference . . . . .	1544
6.274.1 Detailed Description . . . . .	1545
6.274.2 Constructor & Destructor Documentation . . . . .	1545
6.274.2.1 ControlCommandMarshaller . . . . .	1545
6.274.2.2 ~ControlCommandMarshaller . . . . .	1545
6.274.3 Member Function Documentation . . . . .	1545
6.274.3.1 createObject . . . . .	1545
6.274.3.2 getDataStructureType . . . . .	1545
6.274.3.3 looseMarshal . . . . .	1545
6.274.3.4 looseUnmarshal . . . . .	1546
6.274.3.5 tightMarshal1 . . . . .	1546
6.274.3.6 tightMarshal2 . . . . .	1547
6.274.3.7 tightUnmarshal . . . . .	1547
6.275activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller	
Class Reference . . . . .	1548
6.275.1 Detailed Description . . . . .	1549
6.275.2 Constructor & Destructor Documentation . . . . .	1549
6.275.2.1 ControlCommandMarshaller . . . . .	1549
6.275.2.2 ~ControlCommandMarshaller . . . . .	1549
6.275.3 Member Function Documentation . . . . .	1549
6.275.3.1 createObject . . . . .	1549
6.275.3.2 getDataStructureType . . . . .	1549
6.275.3.3 looseMarshal . . . . .	1550
6.275.3.4 looseUnmarshal . . . . .	1550
6.275.3.5 tightMarshal1 . . . . .	1551
6.275.3.6 tightMarshal2 . . . . .	1551
6.275.3.7 tightUnmarshal . . . . .	1552
6.276activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller	
Class Reference . . . . .	1552
6.276.1 Detailed Description . . . . .	1553
6.276.2 Constructor & Destructor Documentation . . . . .	1553

6.276.2.1	ControlCommandMarshaller	1553
6.276.2.2	~ControlCommandMarshaller	1553
6.276.3	Member Function Documentation	1553
6.276.3.1	createObject	1553
6.276.3.2	getDataStructureType	1554
6.276.3.3	looseMarshal	1554
6.276.3.4	looseUnmarshal	1554
6.276.3.5	tightMarshal1	1555
6.276.3.6	tightMarshal2	1555
6.276.3.7	tightUnmarshal	1556
6.277	activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller	
	Class Reference	1556
6.277.1	Detailed Description	1557
6.277.2	Constructor & Destructor Documentation	1558
6.277.2.1	ControlCommandMarshaller	1558
6.277.2.2	~ControlCommandMarshaller	1558
6.277.3	Member Function Documentation	1558
6.277.3.1	createObject	1558
6.277.3.2	getDataStructureType	1558
6.277.3.3	looseMarshal	1558
6.277.3.4	looseUnmarshal	1559
6.277.3.5	tightMarshal1	1559
6.277.3.6	tightMarshal2	1560
6.277.3.7	tightUnmarshal	1560
6.278	activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller	
	Class Reference	1561
6.278.1	Detailed Description	1562
6.278.2	Constructor & Destructor Documentation	1562
6.278.2.1	ControlCommandMarshaller	1562
6.278.2.2	~ControlCommandMarshaller	1562
6.278.3	Member Function Documentation	1562
6.278.3.1	createObject	1562
6.278.3.2	getDataStructureType	1562
6.278.3.3	looseMarshal	1562
6.278.3.4	looseUnmarshal	1563
6.278.3.5	tightMarshal1	1563
6.278.3.6	tightMarshal2	1564
6.278.3.7	tightUnmarshal	1564
6.279	decaf::util::concurrent::CountDownLatch Class Reference	1565
6.279.1	Constructor & Destructor Documentation	1566
6.279.1.1	CountDownLatch	1566
6.279.1.2	~CountDownLatch	1566
6.279.2	Member Function Documentation	1566
6.279.2.1	await	1566
6.279.2.2	await	1566
6.279.2.3	await	1567
6.279.2.4	countDown	1568
6.279.2.5	getCount	1568
6.280	decaf::util::zip::CRC32 Class Reference	1568
6.280.1	Detailed Description	1569

6.280.2 Constructor & Destructor Documentation . . . . .	1569
6.280.2.1 CRC32 . . . . .	1569
6.280.2.2 ~CRC32 . . . . .	1569
6.280.3 Member Function Documentation . . . . .	1569
6.280.3.1 getValue . . . . .	1569
6.280.3.2 reset . . . . .	1569
6.280.3.3 update . . . . .	1569
6.280.3.4 update . . . . .	1570
6.280.3.5 update . . . . .	1570
6.280.3.6 update . . . . .	1570
6.281ct_data_s Struct Reference . . . . .	1571
6.281.1 Field Documentation . . . . .	1571
6.281.1.1 code . . . . .	1571
6.281.1.2 dad . . . . .	1571
6.281.1.3 dl . . . . .	1571
6.281.1.4 fc . . . . .	1571
6.281.1.5 freq . . . . .	1571
6.281.1.6 len . . . . .	1571
6.282activemq::commands::DataArrayResponse Class Reference . . . . .	1572
6.282.1 Constructor & Destructor Documentation . . . . .	1573
6.282.1.1 DataArrayResponse . . . . .	1573
6.282.1.2 ~DataArrayResponse . . . . .	1573
6.282.2 Member Function Documentation . . . . .	1573
6.282.2.1 cloneDataStructure . . . . .	1573
6.282.2.2 copyDataStructure . . . . .	1573
6.282.2.3 equals . . . . .	1573
6.282.2.4 getData . . . . .	1574
6.282.2.5 getData . . . . .	1574
6.282.2.6 getDataStructureType . . . . .	1574
6.282.2.7 setData . . . . .	1574
6.282.2.8 toString . . . . .	1574
6.282.3 Field Documentation . . . . .	1574
6.282.3.1 data . . . . .	1574
6.282.3.2 ID_DATAARRAYRESPONSE . . . . .	1574
6.283activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller	
Class Reference . . . . .	1574
6.283.1 Detailed Description . . . . .	1575
6.283.2 Constructor & Destructor Documentation . . . . .	1576
6.283.2.1 DataArrayResponseMarshaller . . . . .	1576
6.283.2.2 ~DataArrayResponseMarshaller . . . . .	1576
6.283.3 Member Function Documentation . . . . .	1576
6.283.3.1 createObject . . . . .	1576
6.283.3.2 getDataStructureType . . . . .	1576
6.283.3.3 looseMarshal . . . . .	1576
6.283.3.4 looseUnmarshal . . . . .	1577
6.283.3.5 tightMarshal1 . . . . .	1577
6.283.3.6 tightMarshal2 . . . . .	1578
6.283.3.7 tightUnmarshal . . . . .	1578
6.284activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller	
Class Reference . . . . .	1579



6.284.1 Detailed Description . . . . .	1580
6.284.2 Constructor & Destructor Documentation . . . . .	1580
6.284.2.1 DataArrayResponseMarshaller . . . . .	1580
6.284.2.2 ~DataArrayResponseMarshaller . . . . .	1580
6.284.3 Member Function Documentation . . . . .	1580
6.284.3.1 createObject . . . . .	1580
6.284.3.2 getDataStructureType . . . . .	1580
6.284.3.3 looseMarshal . . . . .	1581
6.284.3.4 looseUnmarshal . . . . .	1581
6.284.3.5 tightMarshal1 . . . . .	1581
6.284.3.6 tightMarshal2 . . . . .	1582
6.284.3.7 tightUnmarshal . . . . .	1582
6.285activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller	
Class Reference . . . . .	1583
6.285.1 Detailed Description . . . . .	1584
6.285.2 Constructor & Destructor Documentation . . . . .	1584
6.285.2.1 DataArrayResponseMarshaller . . . . .	1584
6.285.2.2 ~DataArrayResponseMarshaller . . . . .	1584
6.285.3 Member Function Documentation . . . . .	1584
6.285.3.1 createObject . . . . .	1584
6.285.3.2 getDataStructureType . . . . .	1585
6.285.3.3 looseMarshal . . . . .	1585
6.285.3.4 looseUnmarshal . . . . .	1585
6.285.3.5 tightMarshal1 . . . . .	1586
6.285.3.6 tightMarshal2 . . . . .	1586
6.285.3.7 tightUnmarshal . . . . .	1587
6.286activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller	
Class Reference . . . . .	1587
6.286.1 Detailed Description . . . . .	1588
6.286.2 Constructor & Destructor Documentation . . . . .	1589
6.286.2.1 DataArrayResponseMarshaller . . . . .	1589
6.286.2.2 ~DataArrayResponseMarshaller . . . . .	1589
6.286.3 Member Function Documentation . . . . .	1589
6.286.3.1 createObject . . . . .	1589
6.286.3.2 getDataStructureType . . . . .	1589
6.286.3.3 looseMarshal . . . . .	1589
6.286.3.4 looseUnmarshal . . . . .	1590
6.286.3.5 tightMarshal1 . . . . .	1590
6.286.3.6 tightMarshal2 . . . . .	1591
6.286.3.7 tightUnmarshal . . . . .	1591
6.287activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller	
Class Reference . . . . .	1592
6.287.1 Detailed Description . . . . .	1593
6.287.2 Constructor & Destructor Documentation . . . . .	1593
6.287.2.1 DataArrayResponseMarshaller . . . . .	1593
6.287.2.2 ~DataArrayResponseMarshaller . . . . .	1593
6.287.3 Member Function Documentation . . . . .	1593
6.287.3.1 createObject . . . . .	1593
6.287.3.2 getDataStructureType . . . . .	1593
6.287.3.3 looseMarshal . . . . .	1594

6.287.3.4 looseUnmarshal . . . . .	1594
6.287.3.5 tightMarshal1 . . . . .	1594
6.287.3.6 tightMarshal2 . . . . .	1595
6.287.3.7 tightUnmarshal . . . . .	1595
6.288activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller	
Class Reference . . . . .	1596
6.288.1 Detailed Description . . . . .	1597
6.288.2 Constructor & Destructor Documentation . . . . .	1597
6.288.2.1 DataArrayResponseMarshaller . . . . .	1597
6.288.2.2 ~DataArrayResponseMarshaller . . . . .	1597
6.288.3 Member Function Documentation . . . . .	1597
6.288.3.1 createObject . . . . .	1597
6.288.3.2 getDataStructureType . . . . .	1598
6.288.3.3 looseMarshal . . . . .	1598
6.288.3.4 looseUnmarshal . . . . .	1598
6.288.3.5 tightMarshal1 . . . . .	1599
6.288.3.6 tightMarshal2 . . . . .	1599
6.288.3.7 tightUnmarshal . . . . .	1600
6.289decaf::util::zip::DataFormatException Class Reference . . . . .	1600
6.289.1 Constructor & Destructor Documentation . . . . .	1601
6.289.1.1 DataFormatException . . . . .	1601
6.289.1.2 DataFormatException . . . . .	1601
6.289.1.3 DataFormatException . . . . .	1601
6.289.1.4 DataFormatException . . . . .	1602
6.289.1.5 DataFormatException . . . . .	1602
6.289.1.6 DataFormatException . . . . .	1602
6.289.1.7 ~DataFormatException . . . . .	1602
6.289.2 Member Function Documentation . . . . .	1602
6.289.2.1 clone . . . . .	1602
6.290decaf::io::DataInput Class Reference . . . . .	1603
6.290.1 Detailed Description . . . . .	1605
6.290.2 Constructor & Destructor Documentation . . . . .	1605
6.290.2.1 ~DataInput . . . . .	1605
6.290.3 Member Function Documentation . . . . .	1605
6.290.3.1 readBoolean . . . . .	1605
6.290.3.2 readByte . . . . .	1605
6.290.3.3 readChar . . . . .	1606
6.290.3.4 readDouble . . . . .	1606
6.290.3.5 readFloat . . . . .	1607
6.290.3.6 readFully . . . . .	1607
6.290.3.7 readFully . . . . .	1608
6.290.3.8 readInt . . . . .	1608
6.290.3.9 readLine . . . . .	1609
6.290.3.10readLong . . . . .	1609
6.290.3.11readShort . . . . .	1610
6.290.3.12readString . . . . .	1610
6.290.3.13readUnsignedByte . . . . .	1611
6.290.3.14readUnsignedShort . . . . .	1611
6.290.3.15readUTF . . . . .	1611
6.290.3.16skipBytes . . . . .	1612

6.291decaf::io::DataInputStream Class Reference . . . . .	1612
6.291.1 Detailed Description . . . . .	1614
6.291.2 Constructor & Destructor Documentation . . . . .	1615
6.291.2.1 DataInputStream . . . . .	1615
6.291.2.2 ~DataInputStream . . . . .	1615
6.291.3 Member Function Documentation . . . . .	1615
6.291.3.1 readBoolean . . . . .	1615
6.291.3.2 readByte . . . . .	1615
6.291.3.3 readChar . . . . .	1616
6.291.3.4 readDouble . . . . .	1616
6.291.3.5 readFloat . . . . .	1616
6.291.3.6 readFully . . . . .	1617
6.291.3.7 readFully . . . . .	1617
6.291.3.8 readInt . . . . .	1618
6.291.3.9 readLine . . . . .	1618
6.291.3.10readLong . . . . .	1619
6.291.3.11readShort . . . . .	1619
6.291.3.12readString . . . . .	1620
6.291.3.13readUnsignedByte . . . . .	1620
6.291.3.14readUnsignedShort . . . . .	1621
6.291.3.15readUTF . . . . .	1621
6.291.3.16skipBytes . . . . .	1621
6.292decaf::io::DataOutput Class Reference . . . . .	1622
6.292.1 Detailed Description . . . . .	1623
6.292.2 Constructor & Destructor Documentation . . . . .	1624
6.292.2.1 ~DataOutput . . . . .	1624
6.292.3 Member Function Documentation . . . . .	1624
6.292.3.1 writeBoolean . . . . .	1624
6.292.3.2 writeByte . . . . .	1624
6.292.3.3 writeBytes . . . . .	1624
6.292.3.4 writeChar . . . . .	1625
6.292.3.5 writeChars . . . . .	1625
6.292.3.6 writeDouble . . . . .	1625
6.292.3.7 writeFloat . . . . .	1626
6.292.3.8 writeInt . . . . .	1626
6.292.3.9 writeLong . . . . .	1626
6.292.3.10writeShort . . . . .	1627
6.292.3.11writeUnsignedShort . . . . .	1627
6.292.3.12writeUTF . . . . .	1627
6.293decaf::io::DataOutputStream Class Reference . . . . .	1628
6.293.1 Detailed Description . . . . .	1629
6.293.2 Constructor & Destructor Documentation . . . . .	1629
6.293.2.1 DataOutputStream . . . . .	1629
6.293.2.2 ~DataOutputStream . . . . .	1630
6.293.3 Member Function Documentation . . . . .	1630
6.293.3.1 doWriteArrayBounded . . . . .	1630
6.293.3.2 doWriteByte . . . . .	1630
6.293.3.3 size . . . . .	1630
6.293.3.4 writeBoolean . . . . .	1631
6.293.3.5 writeByte . . . . .	1631

6.293.3.6	writeBytes	1631
6.293.3.7	writeChar	1631
6.293.3.8	writeChars	1631
6.293.3.9	writeDouble	1631
6.293.3.10	writeFloat	1631
6.293.3.11	writeInt	1631
6.293.3.12	writeLong	1631
6.293.3.13	writeShort	1631
6.293.3.14	writeUnsignedShort	1631
6.293.3.15	writeUTF	1631
6.293.4	Field Documentation	1631
6.293.4.1	buffer	1631
6.293.4.2	written	1631
6.294	activemq::commands::DataResponse Class Reference	1632
6.294.1	Constructor & Destructor Documentation	1633
6.294.1.1	DataResponse	1633
6.294.1.2	~DataResponse	1633
6.294.2	Member Function Documentation	1633
6.294.2.1	cloneDataStructure	1633
6.294.2.2	copyDataStructure	1633
6.294.2.3	equals	1633
6.294.2.4	getData	1634
6.294.2.5	getData	1634
6.294.2.6	getDataStructureType	1634
6.294.2.7	setData	1634
6.294.2.8	toString	1634
6.294.3	Field Documentation	1634
6.294.3.1	data	1634
6.294.3.2	ID_DATARESPONSE	1634
6.295	activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller	
	Class Reference	1635
6.295.1	Detailed Description	1636
6.295.2	Constructor & Destructor Documentation	1636
6.295.2.1	DataResponseMarshaller	1636
6.295.2.2	~DataResponseMarshaller	1636
6.295.3	Member Function Documentation	1636
6.295.3.1	createObject	1636
6.295.3.2	getDataStructureType	1636
6.295.3.3	looseMarshal	1636
6.295.3.4	looseUnmarshal	1637
6.295.3.5	tightMarshal1	1637
6.295.3.6	tightMarshal2	1638
6.295.3.7	tightUnmarshal	1638
6.296	activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller	
	Class Reference	1639
6.296.1	Detailed Description	1640
6.296.2	Constructor & Destructor Documentation	1640
6.296.2.1	DataResponseMarshaller	1640
6.296.2.2	~DataResponseMarshaller	1640
6.296.3	Member Function Documentation	1640

6.296.3.1	createObject . . . . .	1640
6.296.3.2	getDataStructureType . . . . .	1640
6.296.3.3	looseMarshal . . . . .	1641
6.296.3.4	looseUnmarshal . . . . .	1641
6.296.3.5	tightMarshal1 . . . . .	1642
6.296.3.6	tightMarshal2 . . . . .	1642
6.296.3.7	tightUnmarshal . . . . .	1643
6.297	activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	
	Class Reference . . . . .	1643
6.297.1	Detailed Description . . . . .	1644
6.297.2	Constructor & Destructor Documentation . . . . .	1644
	6.297.2.1 DataResponseMarshaller . . . . .	1644
	6.297.2.2 ~DataResponseMarshaller . . . . .	1644
6.297.3	Member Function Documentation . . . . .	1644
	6.297.3.1 createObject . . . . .	1644
	6.297.3.2 getDataStructureType . . . . .	1645
	6.297.3.3 looseMarshal . . . . .	1645
	6.297.3.4 looseUnmarshal . . . . .	1645
	6.297.3.5 tightMarshal1 . . . . .	1646
	6.297.3.6 tightMarshal2 . . . . .	1646
	6.297.3.7 tightUnmarshal . . . . .	1647
6.298	activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	
	Class Reference . . . . .	1647
6.298.1	Detailed Description . . . . .	1648
6.298.2	Constructor & Destructor Documentation . . . . .	1649
	6.298.2.1 DataResponseMarshaller . . . . .	1649
	6.298.2.2 ~DataResponseMarshaller . . . . .	1649
6.298.3	Member Function Documentation . . . . .	1649
	6.298.3.1 createObject . . . . .	1649
	6.298.3.2 getDataStructureType . . . . .	1649
	6.298.3.3 looseMarshal . . . . .	1649
	6.298.3.4 looseUnmarshal . . . . .	1650
	6.298.3.5 tightMarshal1 . . . . .	1650
	6.298.3.6 tightMarshal2 . . . . .	1651
	6.298.3.7 tightUnmarshal . . . . .	1651
6.299	activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller	
	Class Reference . . . . .	1652
6.299.1	Detailed Description . . . . .	1653
6.299.2	Constructor & Destructor Documentation . . . . .	1653
	6.299.2.1 DataResponseMarshaller . . . . .	1653
	6.299.2.2 ~DataResponseMarshaller . . . . .	1653
6.299.3	Member Function Documentation . . . . .	1653
	6.299.3.1 createObject . . . . .	1653
	6.299.3.2 getDataStructureType . . . . .	1653
	6.299.3.3 looseMarshal . . . . .	1654
	6.299.3.4 looseUnmarshal . . . . .	1654
	6.299.3.5 tightMarshal1 . . . . .	1654
	6.299.3.6 tightMarshal2 . . . . .	1655
	6.299.3.7 tightUnmarshal . . . . .	1655

6.300activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	
Class Reference . . . . .	1656
6.300.1 Detailed Description . . . . .	1657
6.300.2 Constructor & Destructor Documentation . . . . .	1657
6.300.2.1 DataResponseMarshaller . . . . .	1657
6.300.2.2 ~DataResponseMarshaller . . . . .	1657
6.300.3 Member Function Documentation . . . . .	1657
6.300.3.1 createObject . . . . .	1657
6.300.3.2 getDataStructureType . . . . .	1658
6.300.3.3 looseMarshal . . . . .	1658
6.300.3.4 looseUnmarshal . . . . .	1658
6.300.3.5 tightMarshal1 . . . . .	1659
6.300.3.6 tightMarshal2 . . . . .	1659
6.300.3.7 tightUnmarshal . . . . .	1660
6.301activemq::wireformat::openwire::marshal::DataStreamMarshaller Class	
Reference . . . . .	1660
6.301.1 Detailed Description . . . . .	1661
6.301.2 Constructor & Destructor Documentation . . . . .	1661
6.301.2.1 ~DataStreamMarshaller . . . . .	1661
6.301.3 Member Function Documentation . . . . .	1661
6.301.3.1 createObject . . . . .	1661
6.301.3.2 getDataStructureType . . . . .	1668
6.301.3.3 looseMarshal . . . . .	1675
6.301.3.4 looseUnmarshal . . . . .	1683
6.301.3.5 tightMarshal1 . . . . .	1690
6.301.3.6 tightMarshal2 . . . . .	1698
6.301.3.7 tightUnmarshal . . . . .	1706
6.302activemq::commands::DataStructure Class Reference . . . . .	1713
6.302.1 Constructor & Destructor Documentation . . . . .	1714
6.302.1.1 ~DataStructure . . . . .	1714
6.302.2 Member Function Documentation . . . . .	1714
6.302.2.1 cloneDataStructure . . . . .	1714
6.302.2.2 copyDataStructure . . . . .	1715
6.302.2.3 equals . . . . .	1716
6.302.2.4 getDataStructureType . . . . .	1717
6.302.2.5 toString . . . . .	1718
6.303decaf::util::Date Class Reference . . . . .	1719
6.303.1 Detailed Description . . . . .	1720
6.303.2 Constructor & Destructor Documentation . . . . .	1720
6.303.2.1 Date . . . . .	1720
6.303.2.2 Date . . . . .	1720
6.303.2.3 Date . . . . .	1720
6.303.2.4 ~Date . . . . .	1721
6.303.3 Member Function Documentation . . . . .	1721
6.303.3.1 after . . . . .	1721
6.303.3.2 before . . . . .	1721
6.303.3.3 compareTo . . . . .	1721
6.303.3.4 equals . . . . .	1721
6.303.3.5 getTime . . . . .	1722
6.303.3.6 operator< . . . . .	1722

6.303.3.7	operator=	1722
6.303.3.8	operator==	1722
6.303.3.9	setTime	1723
6.303.3.10	toString	1723
6.304	decaf::internal::DecafRuntime Class Reference	1723
6.304.1	Detailed Description	1724
6.304.2	Constructor & Destructor Documentation	1724
6.304.2.1	DecafRuntime	1724
6.304.2.2	~DecafRuntime	1724
6.304.3	Member Function Documentation	1724
6.304.3.1	getGlobalPool	1724
6.305	activemq::threads::DedicatedTaskRunner Class Reference	1724
6.305.1	Constructor & Destructor Documentation	1725
6.305.1.1	DedicatedTaskRunner	1725
6.305.1.2	~DedicatedTaskRunner	1725
6.305.2	Member Function Documentation	1725
6.305.2.1	run	1725
6.305.2.2	shutdown	1725
6.305.2.3	shutdown	1726
6.305.2.4	wakeup	1726
6.306	activemq::core::policies::DefaultPrefetchPolicy Class Reference	1726
6.306.1	Constructor & Destructor Documentation	1727
6.306.1.1	DefaultPrefetchPolicy	1727
6.306.1.2	~DefaultPrefetchPolicy	1727
6.306.2	Member Function Documentation	1727
6.306.2.1	clone	1727
6.306.2.2	getDurableTopicPrefetch	1728
6.306.2.3	getMaxPrefetchLimit	1728
6.306.2.4	getQueueBrowserPrefetch	1728
6.306.2.5	getQueuePrefetch	1728
6.306.2.6	getTopicPrefetch	1729
6.306.2.7	setDurableTopicPrefetch	1729
6.306.2.8	setQueueBrowserPrefetch	1729
6.306.2.9	setQueuePrefetch	1729
6.306.2.10	setTopicPrefetch	1730
6.306.3	Field Documentation	1730
6.306.3.1	DEFAULT_DURABLE_TOPIC_PREFETCH	1730
6.306.3.2	DEFAULT_QUEUE_BROWSER_PREFETCH	1730
6.306.3.3	DEFAULT_QUEUE_PREFETCH	1730
6.306.3.4	DEFAULT_TOPIC_PREFETCH	1730
6.306.3.5	MAX_PREFETCH_SIZE	1730
6.307	activemq::core::policies::DefaultRedeliveryPolicy Class Reference	1730
6.307.1	Constructor & Destructor Documentation	1731
6.307.1.1	DefaultRedeliveryPolicy	1731
6.307.1.2	~DefaultRedeliveryPolicy	1731
6.307.2	Member Function Documentation	1731
6.307.2.1	clone	1731
6.307.2.2	getBackOffMultiplier	1732
6.307.2.3	getCollisionAvoidancePercent	1732
6.307.2.4	getInitialRedeliveryDelay	1732

6.307.2.5	getMaximumRedeliveries	1732
6.307.2.6	getRedeliveryDelay	1732
6.307.2.7	isUseCollisionAvoidance	1733
6.307.2.8	isUseExponentialBackOff	1733
6.307.2.9	setBackOffMultiplier	1733
6.307.2.10	setCollisionAvoidancePercent	1733
6.307.2.11	setInitialRedeliveryDelay	1734
6.307.2.12	setMaximumRedeliveries	1734
6.307.2.13	setUseCollisionAvoidance	1734
6.307.2.14	setUseExponentialBackOff	1734
6.308	decaf::internal::net::DefaultServerSocketFactory Class Reference	1735
6.308.1	Detailed Description	1736
6.308.2	Constructor & Destructor Documentation	1736
6.308.2.1	DefaultServerSocketFactory	1736
6.308.2.2	~DefaultServerSocketFactory	1736
6.308.3	Member Function Documentation	1736
6.308.3.1	createServerSocket	1736
6.308.3.2	createServerSocket	1737
6.308.3.3	createServerSocket	1737
6.308.3.4	createServerSocket	1738
6.309	decaf::internal::net::DefaultSocketFactory Class Reference	1738
6.309.1	Detailed Description	1740
6.309.2	Constructor & Destructor Documentation	1741
6.309.2.1	DefaultSocketFactory	1741
6.309.2.2	~DefaultSocketFactory	1741
6.309.3	Member Function Documentation	1741
6.309.3.1	createSocket	1741
6.309.3.2	createSocket	1741
6.309.3.3	createSocket	1742
6.309.3.4	createSocket	1742
6.309.3.5	createSocket	1743
6.310	decaf::internal::net::ssl::DefaultSSLContext Class Reference	1743
6.310.1	Detailed Description	1744
6.310.2	Constructor & Destructor Documentation	1744
6.310.2.1	DefaultSSLContext	1744
6.310.2.2	~DefaultSSLContext	1744
6.310.3	Member Function Documentation	1744
6.310.3.1	getContext	1744
6.311	decaf::internal::net::ssl::DefaultSSLServerSocketFactory Class Reference	1744
6.311.1	Detailed Description	1746
6.311.2	Constructor & Destructor Documentation	1747
6.311.2.1	DefaultSSLServerSocketFactory	1747
6.311.2.2	~DefaultSSLServerSocketFactory	1747
6.311.3	Member Function Documentation	1747
6.311.3.1	createServerSocket	1747
6.311.3.2	createServerSocket	1747
6.311.3.3	createServerSocket	1748
6.311.3.4	createServerSocket	1748
6.311.3.5	getDefaultCipherSuites	1749



6.311.3.6	getSupportedCipherSuites . . . . .	1749
6.312	decaf::internal::net::ssl::DefaultSSLSocketFactory Class Reference . .	1749
6.312.1	Detailed Description . . . . .	1753
6.312.2	Constructor & Destructor Documentation . . . . .	1753
6.312.2.1	DefaultSSLSocketFactory . . . . .	1753
6.312.2.2	~DefaultSSLSocketFactory . . . . .	1753
6.312.3	Member Function Documentation . . . . .	1753
6.312.3.1	createSocket . . . . .	1753
6.312.3.2	createSocket . . . . .	1753
6.312.3.3	createSocket . . . . .	1754
6.312.3.4	createSocket . . . . .	1755
6.312.3.5	createSocket . . . . .	1755
6.312.3.6	createSocket . . . . .	1756
6.312.3.7	getDefaultCipherSuites . . . . .	1756
6.312.3.8	getSupportedCipherSuites . . . . .	1757
6.313	activemq::transport::DefaultTransportListener Class Reference . . . .	1757
6.313.1	Constructor & Destructor Documentation . . . . .	1758
6.313.1.1	~DefaultTransportListener . . . . .	1758
6.313.2	Member Function Documentation . . . . .	1758
6.313.2.1	onCommand . . . . .	1758
6.313.2.2	onException . . . . .	1758
6.313.2.3	transportInterrupted . . . . .	1758
6.313.2.4	transportResumed . . . . .	1758
6.314	decaf::util::zip::Deflater Class Reference . . . . .	1759
6.314.1	Detailed Description . . . . .	1761
6.314.2	Constructor & Destructor Documentation . . . . .	1761
6.314.2.1	Deflater . . . . .	1761
6.314.2.2	Deflater . . . . .	1762
6.314.2.3	~Deflater . . . . .	1762
6.314.3	Member Function Documentation . . . . .	1762
6.314.3.1	deflate . . . . .	1762
6.314.3.2	deflate . . . . .	1762
6.314.3.3	deflate . . . . .	1763
6.314.3.4	end . . . . .	1763
6.314.3.5	finish . . . . .	1763
6.314.3.6	finished . . . . .	1764
6.314.3.7	getAdler . . . . .	1764
6.314.3.8	getBytesRead . . . . .	1764
6.314.3.9	getBytesWritten . . . . .	1764
6.314.3.10	needsInput . . . . .	1764
6.314.3.11	reset . . . . .	1765
6.314.3.12	setDictionary . . . . .	1765
6.314.3.13	setDictionary . . . . .	1765
6.314.3.14	setDictionary . . . . .	1766
6.314.3.15	setInput . . . . .	1766
6.314.3.16	setInput . . . . .	1767
6.314.3.17	setInput . . . . .	1767
6.314.3.18	setLevel . . . . .	1768
6.314.3.19	setStrategy . . . . .	1768
6.314.4	Field Documentation . . . . .	1768

6.314.4.1 BEST_COMPRESSION . . . . .	1768
6.314.4.2 BEST_SPEED . . . . .	1768
6.314.4.3 DEFAULT_COMPRESSION . . . . .	1768
6.314.4.4 DEFAULT_STRATEGY . . . . .	1769
6.314.4.5 DEFLATED . . . . .	1769
6.314.4.6 FILTERED . . . . .	1769
6.314.4.7 HUFFMAN_ONLY . . . . .	1769
6.314.4.8 NO_COMPRESSION . . . . .	1769
6.315decaf::util::zip::DeflaterOutputStream Class Reference . . . . .	1769
6.315.1 Detailed Description . . . . .	1771
6.315.2 Constructor & Destructor Documentation . . . . .	1771
6.315.2.1 DeflaterOutputStream . . . . .	1771
6.315.2.2 DeflaterOutputStream . . . . .	1771
6.315.2.3 DeflaterOutputStream . . . . .	1772
6.315.2.4 ~DeflaterOutputStream . . . . .	1772
6.315.3 Member Function Documentation . . . . .	1772
6.315.3.1 close . . . . .	1772
6.315.3.2 deflate . . . . .	1773
6.315.3.3 doWriteArrayBounded . . . . .	1773
6.315.3.4 doWriteByte . . . . .	1773
6.315.3.5 finish . . . . .	1773
6.315.4 Field Documentation . . . . .	1773
6.315.4.1 buf . . . . .	1773
6.315.4.2 DEFAULT_BUFFER_SIZE . . . . .	1774
6.315.4.3 deflater . . . . .	1774
6.315.4.4 isDone . . . . .	1774
6.315.4.5 ownDeflater . . . . .	1774
6.316decaf::util::concurrent::Delayed Class Reference . . . . .	1774
6.316.1 Detailed Description . . . . .	1774
6.316.2 Constructor & Destructor Documentation . . . . .	1775
6.316.2.1 ~Delayed . . . . .	1775
6.316.3 Member Function Documentation . . . . .	1775
6.316.3.1 getDelay . . . . .	1775
6.317cms::DeliveryMode Class Reference . . . . .	1775
6.317.1 Detailed Description . . . . .	1775
6.317.2 Member Enumeration Documentation . . . . .	1776
6.317.2.1 DELIVERY_MODE . . . . .	1776
6.317.3 Constructor & Destructor Documentation . . . . .	1776
6.317.3.1 ~DeliveryMode . . . . .	1776
6.318cms::Destination Class Reference . . . . .	1776
6.318.1 Detailed Description . . . . .	1777
6.318.2 Member Enumeration Documentation . . . . .	1777
6.318.2.1 DestinationType . . . . .	1777
6.318.3 Constructor & Destructor Documentation . . . . .	1778
6.318.3.1 ~Destination . . . . .	1778
6.318.4 Member Function Documentation . . . . .	1778
6.318.4.1 clone . . . . .	1778
6.318.4.2 copy . . . . .	1778
6.318.4.3 getCMSProperties . . . . .	1778
6.318.4.4 getDestinationType . . . . .	1779

6.319activemq::commands::ActiveMQDestination::DestinationFilter Struct Reference . . . . .	1779
6.319.1 Field Documentation . . . . .	1779
6.319.1.1 ANY_CHILD . . . . .	1779
6.319.1.2 ANY_DESCENDENT . . . . .	1779
6.320activemq::commands::DestinationInfo Class Reference . . . . .	1779
6.320.1 Constructor & Destructor Documentation . . . . .	1781
6.320.1.1 DestinationInfo . . . . .	1781
6.320.1.2 ~DestinationInfo . . . . .	1781
6.320.2 Member Function Documentation . . . . .	1781
6.320.2.1 cloneDataStructure . . . . .	1781
6.320.2.2 copyDataStructure . . . . .	1781
6.320.2.3 equals . . . . .	1781
6.320.2.4 getBrokerPath . . . . .	1782
6.320.2.5 getBrokerPath . . . . .	1782
6.320.2.6 getConnectionId . . . . .	1782
6.320.2.7 getConnectionId . . . . .	1782
6.320.2.8 getDataStructureType . . . . .	1782
6.320.2.9 getDestination . . . . .	1783
6.320.2.10getDestination . . . . .	1783
6.320.2.11getOperationType . . . . .	1783
6.320.2.12getTimeout . . . . .	1783
6.320.2.13setBrokerPath . . . . .	1783
6.320.2.14setConnectionId . . . . .	1783
6.320.2.15setDestination . . . . .	1783
6.320.2.16setOperationType . . . . .	1783
6.320.2.17setTimeout . . . . .	1783
6.320.2.18toString . . . . .	1783
6.320.2.19visit . . . . .	1783
6.320.3 Field Documentation . . . . .	1784
6.320.3.1 brokerPath . . . . .	1784
6.320.3.2 connectionId . . . . .	1784
6.320.3.3 destination . . . . .	1784
6.320.3.4 ID_DESTINATIONINFO . . . . .	1784
6.320.3.5 operationType . . . . .	1784
6.320.3.6 timeout . . . . .	1784
6.321activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller Class Reference . . . . .	1784
6.321.1 Detailed Description . . . . .	1785
6.321.2 Constructor & Destructor Documentation . . . . .	1786
6.321.2.1 DestinationInfoMarshaller . . . . .	1786
6.321.2.2 ~DestinationInfoMarshaller . . . . .	1786
6.321.3 Member Function Documentation . . . . .	1786
6.321.3.1 createObject . . . . .	1786
6.321.3.2 getDataStructureType . . . . .	1786
6.321.3.3 looseMarshal . . . . .	1786
6.321.3.4 looseUnmarshal . . . . .	1787
6.321.3.5 tightMarshal1 . . . . .	1787
6.321.3.6 tightMarshal2 . . . . .	1788
6.321.3.7 tightUnmarshal . . . . .	1788

6.322activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	
Class Reference . . . . .	1789
6.322.1 Detailed Description . . . . .	1790
6.322.2 Constructor & Destructor Documentation . . . . .	1790
6.322.2.1 DestinationInfoMarshaller . . . . .	1790
6.322.2.2 ~DestinationInfoMarshaller . . . . .	1790
6.322.3 Member Function Documentation . . . . .	1790
6.322.3.1 createObject . . . . .	1790
6.322.3.2 getDataStructureType . . . . .	1790
6.322.3.3 looseMarshal . . . . .	1790
6.322.3.4 looseUnmarshal . . . . .	1791
6.322.3.5 tightMarshal1 . . . . .	1791
6.322.3.6 tightMarshal2 . . . . .	1792
6.322.3.7 tightUnmarshal . . . . .	1792
6.323activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller	
Class Reference . . . . .	1793
6.323.1 Detailed Description . . . . .	1794
6.323.2 Constructor & Destructor Documentation . . . . .	1794
6.323.2.1 DestinationInfoMarshaller . . . . .	1794
6.323.2.2 ~DestinationInfoMarshaller . . . . .	1794
6.323.3 Member Function Documentation . . . . .	1794
6.323.3.1 createObject . . . . .	1794
6.323.3.2 getDataStructureType . . . . .	1794
6.323.3.3 looseMarshal . . . . .	1795
6.323.3.4 looseUnmarshal . . . . .	1795
6.323.3.5 tightMarshal1 . . . . .	1796
6.323.3.6 tightMarshal2 . . . . .	1796
6.323.3.7 tightUnmarshal . . . . .	1797
6.324activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	
Class Reference . . . . .	1797
6.324.1 Detailed Description . . . . .	1798
6.324.2 Constructor & Destructor Documentation . . . . .	1798
6.324.2.1 DestinationInfoMarshaller . . . . .	1798
6.324.2.2 ~DestinationInfoMarshaller . . . . .	1798
6.324.3 Member Function Documentation . . . . .	1798
6.324.3.1 createObject . . . . .	1798
6.324.3.2 getDataStructureType . . . . .	1799
6.324.3.3 looseMarshal . . . . .	1799
6.324.3.4 looseUnmarshal . . . . .	1799
6.324.3.5 tightMarshal1 . . . . .	1800
6.324.3.6 tightMarshal2 . . . . .	1800
6.324.3.7 tightUnmarshal . . . . .	1801
6.325activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller	
Class Reference . . . . .	1801
6.325.1 Detailed Description . . . . .	1802
6.325.2 Constructor & Destructor Documentation . . . . .	1803
6.325.2.1 DestinationInfoMarshaller . . . . .	1803
6.325.2.2 ~DestinationInfoMarshaller . . . . .	1803
6.325.3 Member Function Documentation . . . . .	1803
6.325.3.1 createObject . . . . .	1803

6.325.3.2	getDataStructureType . . . . .	1803
6.325.3.3	looseMarshal . . . . .	1803
6.325.3.4	looseUnmarshal . . . . .	1804
6.325.3.5	tightMarshal1 . . . . .	1804
6.325.3.6	tightMarshal2 . . . . .	1805
6.325.3.7	tightUnmarshal . . . . .	1805
6.326	activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller	
	Class Reference . . . . .	1806
6.326.1	Detailed Description . . . . .	1807
6.326.2	Constructor & Destructor Documentation . . . . .	1807
6.326.2.1	DestinationInfoMarshaller . . . . .	1807
6.326.2.2	~DestinationInfoMarshaller . . . . .	1807
6.326.3	Member Function Documentation . . . . .	1807
6.326.3.1	createObject . . . . .	1807
6.326.3.2	getDataStructureType . . . . .	1807
6.326.3.3	looseMarshal . . . . .	1807
6.326.3.4	looseUnmarshal . . . . .	1808
6.326.3.5	tightMarshal1 . . . . .	1808
6.326.3.6	tightMarshal2 . . . . .	1809
6.326.3.7	tightUnmarshal . . . . .	1809
6.327	activemq::cmsutil::DestinationResolver Class Reference . . . . .	1810
6.327.1	Detailed Description . . . . .	1810
6.327.2	Constructor & Destructor Documentation . . . . .	1811
6.327.2.1	~DestinationResolver . . . . .	1811
6.327.3	Member Function Documentation . . . . .	1811
6.327.3.1	destroy . . . . .	1811
6.327.3.2	init . . . . .	1811
6.327.3.3	resolveDestinationName . . . . .	1811
6.328	activemq::commands::DiscoveryEvent Class Reference . . . . .	1812
6.328.1	Constructor & Destructor Documentation . . . . .	1813
6.328.1.1	DiscoveryEvent . . . . .	1813
6.328.1.2	~DiscoveryEvent . . . . .	1813
6.328.2	Member Function Documentation . . . . .	1813
6.328.2.1	cloneDataStructure . . . . .	1813
6.328.2.2	copyDataStructure . . . . .	1813
6.328.2.3	equals . . . . .	1813
6.328.2.4	getBrokerName . . . . .	1814
6.328.2.5	getBrokerName . . . . .	1814
6.328.2.6	getDataStructureType . . . . .	1814
6.328.2.7	getServiceName . . . . .	1814
6.328.2.8	getServiceName . . . . .	1814
6.328.2.9	setBrokerName . . . . .	1814
6.328.2.10	setServiceName . . . . .	1814
6.328.2.11	toString . . . . .	1814
6.328.3	Field Documentation . . . . .	1815
6.328.3.1	brokerName . . . . .	1815
6.328.3.2	ID_DISCOVERYEVENT . . . . .	1815
6.328.3.3	serviceName . . . . .	1815
6.329	activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller	
	Class Reference . . . . .	1815

6.329.1 Detailed Description . . . . .	1816
6.329.2 Constructor & Destructor Documentation . . . . .	1816
6.329.2.1 DiscoveryEventMarshaller . . . . .	1816
6.329.2.2 ~DiscoveryEventMarshaller . . . . .	1816
6.329.3 Member Function Documentation . . . . .	1816
6.329.3.1 createObject . . . . .	1816
6.329.3.2 getDataStructureType . . . . .	1816
6.329.3.3 looseMarshal . . . . .	1817
6.329.3.4 looseUnmarshal . . . . .	1817
6.329.3.5 tightMarshal1 . . . . .	1818
6.329.3.6 tightMarshal2 . . . . .	1818
6.329.3.7 tightUnmarshal . . . . .	1819
6.330activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller	
Class Reference . . . . .	1819
6.330.1 Detailed Description . . . . .	1820
6.330.2 Constructor & Destructor Documentation . . . . .	1820
6.330.2.1 DiscoveryEventMarshaller . . . . .	1820
6.330.2.2 ~DiscoveryEventMarshaller . . . . .	1820
6.330.3 Member Function Documentation . . . . .	1820
6.330.3.1 createObject . . . . .	1820
6.330.3.2 getDataStructureType . . . . .	1821
6.330.3.3 looseMarshal . . . . .	1821
6.330.3.4 looseUnmarshal . . . . .	1821
6.330.3.5 tightMarshal1 . . . . .	1822
6.330.3.6 tightMarshal2 . . . . .	1822
6.330.3.7 tightUnmarshal . . . . .	1823
6.331activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller	
Class Reference . . . . .	1823
6.331.1 Detailed Description . . . . .	1824
6.331.2 Constructor & Destructor Documentation . . . . .	1824
6.331.2.1 DiscoveryEventMarshaller . . . . .	1824
6.331.2.2 ~DiscoveryEventMarshaller . . . . .	1824
6.331.3 Member Function Documentation . . . . .	1824
6.331.3.1 createObject . . . . .	1824
6.331.3.2 getDataStructureType . . . . .	1825
6.331.3.3 looseMarshal . . . . .	1825
6.331.3.4 looseUnmarshal . . . . .	1825
6.331.3.5 tightMarshal1 . . . . .	1826
6.331.3.6 tightMarshal2 . . . . .	1826
6.331.3.7 tightUnmarshal . . . . .	1827
6.332activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller	
Class Reference . . . . .	1827
6.332.1 Detailed Description . . . . .	1828
6.332.2 Constructor & Destructor Documentation . . . . .	1828
6.332.2.1 DiscoveryEventMarshaller . . . . .	1828
6.332.2.2 ~DiscoveryEventMarshaller . . . . .	1828
6.332.3 Member Function Documentation . . . . .	1828
6.332.3.1 createObject . . . . .	1828
6.332.3.2 getDataStructureType . . . . .	1829
6.332.3.3 looseMarshal . . . . .	1829

6.332.3.4	looseUnmarshal . . . . .	1829
6.332.3.5	tightMarshal1 . . . . .	1830
6.332.3.6	tightMarshal2 . . . . .	1830
6.332.3.7	tightUnmarshal . . . . .	1831
6.333	activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller	
	Class Reference . . . . .	1831
6.333.1	Detailed Description . . . . .	1832
6.333.2	Constructor & Destructor Documentation . . . . .	1832
6.333.2.1	DiscoveryEventMarshaller . . . . .	1832
6.333.2.2	~DiscoveryEventMarshaller . . . . .	1832
6.333.3	Member Function Documentation . . . . .	1832
6.333.3.1	createObject . . . . .	1832
6.333.3.2	getDataStructureType . . . . .	1833
6.333.3.3	looseMarshal . . . . .	1833
6.333.3.4	looseUnmarshal . . . . .	1833
6.333.3.5	tightMarshal1 . . . . .	1834
6.333.3.6	tightMarshal2 . . . . .	1834
6.333.3.7	tightUnmarshal . . . . .	1835
6.334	activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller	
	Class Reference . . . . .	1835
6.334.1	Detailed Description . . . . .	1836
6.334.2	Constructor & Destructor Documentation . . . . .	1836
6.334.2.1	DiscoveryEventMarshaller . . . . .	1836
6.334.2.2	~DiscoveryEventMarshaller . . . . .	1836
6.334.3	Member Function Documentation . . . . .	1836
6.334.3.1	createObject . . . . .	1836
6.334.3.2	getDataStructureType . . . . .	1837
6.334.3.3	looseMarshal . . . . .	1837
6.334.3.4	looseUnmarshal . . . . .	1837
6.334.3.5	tightMarshal1 . . . . .	1838
6.334.3.6	tightMarshal2 . . . . .	1838
6.334.3.7	tightUnmarshal . . . . .	1839
6.335	activemq::core::DispatchData Class Reference . . . . .	1839
6.335.1	Detailed Description . . . . .	1839
6.335.2	Constructor & Destructor Documentation . . . . .	1840
6.335.2.1	DispatchData . . . . .	1840
6.335.2.2	DispatchData . . . . .	1840
6.335.3	Member Function Documentation . . . . .	1840
6.335.3.1	getConsumerId . . . . .	1840
6.335.3.2	getMessage . . . . .	1840
6.336	activemq::core::Dispatcher Class Reference . . . . .	1840
6.336.1	Detailed Description . . . . .	1840
6.336.2	Constructor & Destructor Documentation . . . . .	1841
6.336.2.1	~Dispatcher . . . . .	1841
6.336.3	Member Function Documentation . . . . .	1841
6.336.3.1	dispatch . . . . .	1841
6.337	decaf::lang::Double Class Reference . . . . .	1841
6.337.1	Constructor & Destructor Documentation . . . . .	1844
6.337.1.1	Double . . . . .	1844
6.337.1.2	Double . . . . .	1844

6.337.1.3 ~Double . . . . .	1844
6.337.2 Member Function Documentation . . . . .	1844
6.337.2.1 byteValue . . . . .	1844
6.337.2.2 compare . . . . .	1844
6.337.2.3 compareTo . . . . .	1845
6.337.2.4 compareTo . . . . .	1845
6.337.2.5 doubleToLongBits . . . . .	1845
6.337.2.6 doubleToRawLongBits . . . . .	1846
6.337.2.7 doubleValue . . . . .	1846
6.337.2.8 equals . . . . .	1847
6.337.2.9 equals . . . . .	1847
6.337.2.10 floatValue . . . . .	1847
6.337.2.11 intValue . . . . .	1847
6.337.2.12 isInfinite . . . . .	1847
6.337.2.13 isInfinite . . . . .	1848
6.337.2.14 isNaN . . . . .	1848
6.337.2.15 isNaN . . . . .	1848
6.337.2.16 longBitsToDouble . . . . .	1848
6.337.2.17 longValue . . . . .	1849
6.337.2.18 operator< . . . . .	1849
6.337.2.19 operator< . . . . .	1849
6.337.2.20 operator== . . . . .	1849
6.337.2.21 operator== . . . . .	1850
6.337.2.22 parseDouble . . . . .	1850
6.337.2.23 shortValue . . . . .	1850
6.337.2.24 toHexString . . . . .	1851
6.337.2.25 toString . . . . .	1851
6.337.2.26 toString . . . . .	1851
6.337.2.27 valueOf . . . . .	1852
6.337.2.28 valueOf . . . . .	1852
6.337.3 Field Documentation . . . . .	1853
6.337.3.1 MAX_VALUE . . . . .	1853
6.337.3.2 MIN_VALUE . . . . .	1853
6.337.3.3 NaN . . . . .	1853
6.337.3.4 NEGATIVE_INFINITY . . . . .	1853
6.337.3.5 POSITIVE_INFINITY . . . . .	1853
6.337.3.6 SIZE . . . . .	1853
6.338 decaf::internal::nio::DoubleArrayBuffer Class Reference . . . . .	1853
6.338.1 Constructor & Destructor Documentation . . . . .	1858
6.338.1.1 DoubleArrayBuffer . . . . .	1858
6.338.1.2 DoubleArrayBuffer . . . . .	1858
6.338.1.3 DoubleArrayBuffer . . . . .	1859
6.338.1.4 DoubleArrayBuffer . . . . .	1859
6.338.1.5 ~DoubleArrayBuffer . . . . .	1859
6.338.2 Member Function Documentation . . . . .	1859
6.338.2.1 array . . . . .	1859
6.338.2.2 arrayOffset . . . . .	1860
6.338.2.3 asReadOnlyBuffer . . . . .	1860
6.338.2.4 compact . . . . .	1861
6.338.2.5 duplicate . . . . .	1861



6.338.2.6	get	1862
6.338.2.7	get	1862
6.338.2.8	hasArray	1863
6.338.2.9	isReadOnly	1863
6.338.2.10	put	1863
6.338.2.11	put	1864
6.338.2.12	setReadOnly	1864
6.338.2.13	slice	1864
6.339	decaf::nio::DoubleBuffer Class Reference	1865
6.339.1	Detailed Description	1867
6.339.2	Constructor & Destructor Documentation	1868
6.339.2.1	DoubleBuffer	1868
6.339.2.2	~DoubleBuffer	1868
6.339.3	Member Function Documentation	1868
6.339.3.1	allocate	1868
6.339.3.2	array	1868
6.339.3.3	arrayOffset	1869
6.339.3.4	asReadOnlyBuffer	1869
6.339.3.5	compact	1870
6.339.3.6	compareTo	1870
6.339.3.7	duplicate	1870
6.339.3.8	equals	1871
6.339.3.9	get	1871
6.339.3.10	get	1871
6.339.3.11	iget	1872
6.339.3.12	iget	1872
6.339.3.13	hasArray	1873
6.339.3.14	operator<	1873
6.339.3.15	operator==	1873
6.339.3.16	put	1873
6.339.3.17	put	1874
6.339.3.18	put	1874
6.339.3.19	put	1875
6.339.3.20	put	1876
6.339.3.21	slice	1876
6.339.3.22	toString	1877
6.339.3.23	wrap	1877
6.339.3.24	wrap	1877
6.340	decaf::lang::DYNAMIC_CAST_TOKEN Struct Reference	1878
6.341	activemq::cmsutil::DynamicDestinationResolver Class Reference	1878
6.341.1	Detailed Description	1879
6.341.2	Constructor & Destructor Documentation	1879
6.341.2.1	DynamicDestinationResolver	1879
6.341.2.2	DynamicDestinationResolver	1879
6.341.2.3	~DynamicDestinationResolver	1879
6.341.3	Member Function Documentation	1879
6.341.3.1	destroy	1879
6.341.3.2	init	1879
6.341.3.3	operator=	1880
6.341.3.4	resolveDestinationName	1880

6.342	decaf::util::Map< K, V, COMPARATOR >::Entry Class Reference	1880
6.342.1	Constructor & Destructor Documentation	1881
6.342.1.1	Entry	1881
6.342.1.2	~Entry	1881
6.342.2	Member Function Documentation	1881
6.342.2.1	getKey	1881
6.342.2.2	getValue	1881
6.342.2.3	setValue	1881
6.343	decaf::io::EOFException Class Reference	1881
6.343.1	Constructor & Destructor Documentation	1882
6.343.1.1	EOFException	1882
6.343.1.2	EOFException	1882
6.343.1.3	EOFException	1882
6.343.1.4	EOFException	1883
6.343.1.5	EOFException	1883
6.343.1.6	EOFException	1883
6.343.1.7	~EOFException	1883
6.343.2	Member Function Documentation	1883
6.343.2.1	clone	1883
6.344	decaf::util::logging::ErrorManager Class Reference	1884
6.344.1	Detailed Description	1885
6.344.2	Constructor & Destructor Documentation	1885
6.344.2.1	ErrorManager	1885
6.344.2.2	~ErrorManager	1885
6.344.3	Member Function Documentation	1885
6.344.3.1	error	1885
6.344.4	Field Documentation	1885
6.344.4.1	CLOSE_FAILURE	1885
6.344.4.2	FLUSH_FAILURE	1886
6.344.4.3	FORMAT_FAILURE	1886
6.344.4.4	GENERIC_FAILURE	1886
6.344.4.5	OPEN_FAILURE	1886
6.344.4.6	WRITE_FAILURE	1886
6.345	decaf::lang::Exception Class Reference	1886
6.345.1	Constructor & Destructor Documentation	1888
6.345.1.1	Exception	1888
6.345.1.2	Exception	1888
6.345.1.3	Exception	1888
6.345.1.4	Exception	1889
6.345.1.5	Exception	1889
6.345.1.6	~Exception	1889
6.345.2	Member Function Documentation	1889
6.345.2.1	buildMessage	1889
6.345.2.2	clone	1889
6.345.2.3	getCause	1890
6.345.2.4	getMessage	1891
6.345.2.5	getStackTrace	1891
6.345.2.6	getStackTraceString	1891
6.345.2.7	initCause	1891
6.345.2.8	operator=	1892

6.345.2.9	printStackTrace . . . . .	1892
6.345.2.10	printStackTrace . . . . .	1892
6.345.2.11	setMark . . . . .	1892
6.345.2.12	setMessage . . . . .	1892
6.345.2.13	setStackTrace . . . . .	1893
6.345.2.14	what . . . . .	1893
6.345.3	Field Documentation . . . . .	1893
6.345.3.1	cause . . . . .	1893
6.345.3.2	message . . . . .	1893
6.345.3.3	stackTrace . . . . .	1893
6.346	cms::ExceptionListener Class Reference . . . . .	1893
6.346.1	Detailed Description . . . . .	1894
6.346.2	Constructor & Destructor Documentation . . . . .	1894
6.346.2.1	~ExceptionListener . . . . .	1894
6.346.3	Member Function Documentation . . . . .	1894
6.346.3.1	onException . . . . .	1894
6.347	activemq::commands::ExceptionResponse Class Reference . . . . .	1894
6.347.1	Constructor & Destructor Documentation . . . . .	1896
6.347.1.1	ExceptionResponse . . . . .	1896
6.347.1.2	~ExceptionResponse . . . . .	1896
6.347.2	Member Function Documentation . . . . .	1896
6.347.2.1	cloneDataStructure . . . . .	1896
6.347.2.2	copyDataStructure . . . . .	1896
6.347.2.3	equals . . . . .	1896
6.347.2.4	getDataStructureType . . . . .	1897
6.347.2.5	getException . . . . .	1897
6.347.2.6	getException . . . . .	1897
6.347.2.7	setException . . . . .	1897
6.347.2.8	toString . . . . .	1897
6.347.3	Field Documentation . . . . .	1897
6.347.3.1	exception . . . . .	1897
6.347.3.2	ID_EXCEPTIONRESPONSE . . . . .	1897
6.348	activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller Class Reference . . . . .	1898
6.348.1	Detailed Description . . . . .	1899
6.348.2	Constructor & Destructor Documentation . . . . .	1899
6.348.2.1	ExceptionResponseMarshaller . . . . .	1899
6.348.2.2	~ExceptionResponseMarshaller . . . . .	1899
6.348.3	Member Function Documentation . . . . .	1899
6.348.3.1	createObject . . . . .	1899
6.348.3.2	getDataStructureType . . . . .	1899
6.348.3.3	looseMarshal . . . . .	1899
6.348.3.4	looseUnmarshal . . . . .	1900
6.348.3.5	tightMarshal1 . . . . .	1900
6.348.3.6	tightMarshal2 . . . . .	1901
6.348.3.7	tightUnmarshal . . . . .	1901
6.349	activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller Class Reference . . . . .	1902
6.349.1	Detailed Description . . . . .	1903
6.349.2	Constructor & Destructor Documentation . . . . .	1903

6.349.2.1	ExceptionResponseMarshaller	1903
6.349.2.2	~ExceptionResponseMarshaller	1903
6.349.3	Member Function Documentation	1903
6.349.3.1	createObject	1903
6.349.3.2	getDataStructureType	1903
6.349.3.3	looseMarshal	1904
6.349.3.4	looseUnmarshal	1904
6.349.3.5	tightMarshal1	1905
6.349.3.6	tightMarshal2	1905
6.349.3.7	tightUnmarshal	1906
6.350	activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	
	Class Reference	1906
6.350.1	Detailed Description	1907
6.350.2	Constructor & Destructor Documentation	1907
6.350.2.1	ExceptionResponseMarshaller	1907
6.350.2.2	~ExceptionResponseMarshaller	1907
6.350.3	Member Function Documentation	1907
6.350.3.1	createObject	1907
6.350.3.2	getDataStructureType	1908
6.350.3.3	looseMarshal	1908
6.350.3.4	looseUnmarshal	1908
6.350.3.5	tightMarshal1	1909
6.350.3.6	tightMarshal2	1909
6.350.3.7	tightUnmarshal	1910
6.351	activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller	
	Class Reference	1910
6.351.1	Detailed Description	1911
6.351.2	Constructor & Destructor Documentation	1912
6.351.2.1	ExceptionResponseMarshaller	1912
6.351.2.2	~ExceptionResponseMarshaller	1912
6.351.3	Member Function Documentation	1912
6.351.3.1	createObject	1912
6.351.3.2	getDataStructureType	1912
6.351.3.3	looseMarshal	1912
6.351.3.4	looseUnmarshal	1913
6.351.3.5	tightMarshal1	1913
6.351.3.6	tightMarshal2	1914
6.351.3.7	tightUnmarshal	1914
6.352	activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller	
	Class Reference	1915
6.352.1	Detailed Description	1916
6.352.2	Constructor & Destructor Documentation	1916
6.352.2.1	ExceptionResponseMarshaller	1916
6.352.2.2	~ExceptionResponseMarshaller	1916
6.352.3	Member Function Documentation	1916
6.352.3.1	createObject	1916
6.352.3.2	getDataStructureType	1916
6.352.3.3	looseMarshal	1917
6.352.3.4	looseUnmarshal	1917
6.352.3.5	tightMarshal1	1917

6.352.3.6	tightMarshal2	1918
6.352.3.7	tightUnmarshal	1918
6.353	activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	
	Class Reference	1919
6.353.1	Detailed Description	1920
6.353.2	Constructor & Destructor Documentation	1920
6.353.2.1	ExceptionResponseMarshaller	1920
6.353.2.2	~ExceptionResponseMarshaller	1920
6.353.3	Member Function Documentation	1920
6.353.3.1	createObject	1920
6.353.3.2	getDataStructureType	1921
6.353.3.3	looseMarshal	1921
6.353.3.4	looseUnmarshal	1921
6.353.3.5	tightMarshal1	1922
6.353.3.6	tightMarshal2	1922
6.353.3.7	tightUnmarshal	1923
6.354	decaf::util::concurrent::ExecutionException Class Reference	1923
6.354.1	Constructor & Destructor Documentation	1924
6.354.1.1	ExecutionException	1924
6.354.1.2	ExecutionException	1924
6.354.1.3	ExecutionException	1924
6.354.1.4	ExecutionException	1925
6.354.1.5	ExecutionException	1925
6.354.1.6	ExecutionException	1925
6.354.1.7	~ExecutionException	1926
6.354.2	Member Function Documentation	1926
6.354.2.1	clone	1926
6.355	decaf::util::concurrent::Executor Class Reference	1926
6.355.1	Detailed Description	1926
6.355.2	Constructor & Destructor Documentation	1928
6.355.2.1	~Executor	1928
6.355.3	Member Function Documentation	1928
6.355.3.1	execute	1928
6.356	decaf::util::concurrent::ExecutorService Class Reference	1928
6.356.1	Detailed Description	1929
6.356.2	Constructor & Destructor Documentation	1929
6.356.2.1	~ExecutorService	1929
6.356.3	Member Function Documentation	1929
6.356.3.1	awaitTermination	1929
6.357	activemq::transport::failover::FailoverTransport Class Reference	1930
6.357.1	Constructor & Destructor Documentation	1933
6.357.1.1	FailoverTransport	1933
6.357.1.2	~FailoverTransport	1933
6.357.2	Member Function Documentation	1933
6.357.2.1	add	1933
6.357.2.2	addURI	1933
6.357.2.3	close	1934
6.357.2.4	getBackOffMultiplier	1934
6.357.2.5	getBackupPoolSize	1934
6.357.2.6	getInitialReconnectDelay	1934

6.357.2.7	getMaxCacheSize	1934
6.357.2.8	getMaxReconnectAttempts	1934
6.357.2.9	getMaxReconnectDelay	1934
6.357.2.10	getReconnectDelay	1934
6.357.2.11	getRemoteAddress	1934
6.357.2.12	getStartupMaxReconnectAttempts	1935
6.357.2.13	getTimeout	1935
6.357.2.14	getTransportListener	1935
6.357.2.15	handleTransportFailure	1935
6.357.2.16	isBackup	1935
6.357.2.17	isClosed	1935
6.357.2.18	isConnected	1936
6.357.2.19	isFaultTolerant	1936
6.357.2.20	isInitialized	1936
6.357.2.21	isPending	1936
6.357.2.22	isRandomize	1937
6.357.2.23	isTrackMessages	1937
6.357.2.24	isTrackTransactionProducers	1937
6.357.2.25	isUseExponentialBackOff	1937
6.357.2.26	iterate	1937
6.357.2.27	narrow	1937
6.357.2.28	oneway	1937
6.357.2.29	reconnect	1938
6.357.2.30	reconnect	1938
6.357.2.31	removeURI	1938
6.357.2.32	request	1939
6.357.2.33	request	1939
6.357.2.34	restoreTransport	1940
6.357.2.35	setBackOffMultiplier	1940
6.357.2.36	setBackup	1940
6.357.2.37	setBackupPoolSize	1940
6.357.2.38	setConnectionInterruptProcessingComplete	1940
6.357.2.39	setInitialized	1940
6.357.2.40	setInitialReconnectDelay	1941
6.357.2.41	setMaxCacheSize	1941
6.357.2.42	setMaxReconnectAttempts	1941
6.357.2.43	setMaxReconnectDelay	1941
6.357.2.44	setRandomize	1941
6.357.2.45	setReconnectDelay	1941
6.357.2.46	setStartupMaxReconnectAttempts	1941
6.357.2.47	setTimeout	1941
6.357.2.48	setTrackMessages	1941
6.357.2.49	setTrackTransactionProducers	1941
6.357.2.50	setTransportListener	1941
6.357.2.51	setUseExponentialBackOff	1941
6.357.2.52	setWireFormat	1941
6.357.2.53	start	1942
6.357.2.54	stop	1942
6.357.3	Friends And Related Function Documentation	1942
6.357.3.1	FailoverTransportListener	1942

6.358activemq::transport::failover::FailoverTransportFactory Class Reference	1942
6.358.1 Detailed Description	1943
6.358.2 Constructor & Destructor Documentation	1944
6.358.2.1 ~FailoverTransportFactory	1944
6.358.3 Member Function Documentation	1944
6.358.3.1 create	1944
6.358.3.2 createComposite	1944
6.358.3.3 doCreateComposite	1945
6.359activemq::transport::failover::FailoverTransportListener Class Reference	1945
6.359.1 Detailed Description	1946
6.359.2 Constructor & Destructor Documentation	1946
6.359.2.1 FailoverTransportListener	1946
6.359.2.2 ~FailoverTransportListener	1946
6.359.3 Member Function Documentation	1946
6.359.3.1 onCommand	1946
6.359.3.2 onException	1946
6.359.3.3 transportInterrupted	1947
6.359.3.4 transportResumed	1947
6.360decaf::io::FileDescriptor Class Reference	1947
6.360.1 Detailed Description	1948
6.360.2 Constructor & Destructor Documentation	1948
6.360.2.1 FileDescriptor	1948
6.360.2.2 FileDescriptor	1948
6.360.2.3 ~FileDescriptor	1948
6.360.3 Member Function Documentation	1948
6.360.3.1 sync	1948
6.360.3.2 valid	1949
6.360.4 Field Documentation	1949
6.360.4.1 descriptor	1949
6.360.4.2 err	1949
6.360.4.3 in	1949
6.360.4.4 out	1949
6.360.4.5 readonly	1949
6.361decaf::util::logging::Filter Class Reference	1949
6.361.1 Detailed Description	1950
6.361.2 Constructor & Destructor Documentation	1950
6.361.2.1 ~Filter	1950
6.361.3 Member Function Documentation	1950
6.361.3.1 isLoggable	1950
6.362decaf::io::FilterInputStream Class Reference	1950
6.362.1 Detailed Description	1953
6.362.2 Constructor & Destructor Documentation	1953
6.362.2.1 FilterInputStream	1953
6.362.2.2 ~FilterInputStream	1953
6.362.3 Member Function Documentation	1953
6.362.3.1 available	1953
6.362.3.2 close	1954
6.362.3.3 doReadArray	1954
6.362.3.4 doReadArrayBounded	1954
6.362.3.5 doReadByte	1954

6.362.3.6	isClosed	1955
6.362.3.7	mark	1955
6.362.3.8	markSupported	1955
6.362.3.9	reset	1956
6.362.3.10	skip	1956
6.362.4	Field Documentation	1957
6.362.4.1	closed	1957
6.362.4.2	inputStream	1957
6.362.4.3	own	1957
6.363	decaf::io::FilterOutputStream Class Reference	1957
6.363.1	Detailed Description	1958
6.363.2	Constructor & Destructor Documentation	1959
6.363.2.1	FilterOutputStream	1959
6.363.2.2	~FilterOutputStream	1959
6.363.3	Member Function Documentation	1959
6.363.3.1	close	1959
6.363.3.2	doWriteArray	1960
6.363.3.3	doWriteArrayBounded	1960
6.363.3.4	doWriteByte	1960
6.363.3.5	flush	1960
6.363.3.6	isClosed	1961
6.363.3.7	toString	1961
6.363.4	Field Documentation	1961
6.363.4.1	closed	1961
6.363.4.2	outputStream	1961
6.363.4.3	own	1961
6.364	decaf::lang::Float Class Reference	1961
6.364.1	Constructor & Destructor Documentation	1964
6.364.1.1	Float	1964
6.364.1.2	Float	1964
6.364.1.3	Float	1964
6.364.1.4	~Float	1964
6.364.2	Member Function Documentation	1964
6.364.2.1	byteValue	1964
6.364.2.2	compare	1965
6.364.2.3	compareTo	1965
6.364.2.4	compareTo	1965
6.364.2.5	doubleValue	1966
6.364.2.6	equals	1966
6.364.2.7	equals	1966
6.364.2.8	floatToIntBits	1966
6.364.2.9	floatToRawIntBits	1967
6.364.2.10	floatValue	1967
6.364.2.11	intBitsToFloat	1967
6.364.2.12	intValue	1968
6.364.2.13	isInfinite	1968
6.364.2.14	isInfinite	1968
6.364.2.15	isNaN	1968
6.364.2.16	isNaN	1969
6.364.2.17	longValue	1969



6.364.2.18	operator<	1969
6.364.2.19	operator<	1969
6.364.2.20	operator==	1970
6.364.2.21	operator==	1970
6.364.2.22	parseFloat	1970
6.364.2.23	shortValue	1971
6.364.2.24	toHexString	1971
6.364.2.25	toString	1972
6.364.2.26	toString	1972
6.364.2.27	valueOf	1972
6.364.2.28	valueOf	1973
6.364.3	Field Documentation	1973
6.364.3.1	MAX_VALUE	1973
6.364.3.2	MIN_VALUE	1973
6.364.3.3	NaN	1973
6.364.3.4	NEGATIVE_INFINITY	1973
6.364.3.5	POSITIVE_INFINITY	1973
6.364.3.6	SIZE	1973
6.365	decaf::nio::FloatArrayBuffer Class Reference	1974
6.365.1	Constructor & Destructor Documentation	1978
6.365.1.1	FloatArrayBuffer	1978
6.365.1.2	FloatArrayBuffer	1978
6.365.1.3	FloatArrayBuffer	1979
6.365.1.4	FloatArrayBuffer	1979
6.365.1.5	~FloatArrayBuffer	1980
6.365.2	Member Function Documentation	1980
6.365.2.1	array	1980
6.365.2.2	arrayOffset	1980
6.365.2.3	asReadOnlyBuffer	1981
6.365.2.4	compact	1981
6.365.2.5	duplicate	1982
6.365.2.6	get	1982
6.365.2.7	get	1982
6.365.2.8	hasArray	1983
6.365.2.9	isReadOnly	1983
6.365.2.10	put	1983
6.365.2.11	put	1984
6.365.2.12	setReadOnly	1984
6.365.2.13	slice	1985
6.366	decaf::nio::FloatBuffer Class Reference	1985
6.366.1	Detailed Description	1987
6.366.2	Constructor & Destructor Documentation	1988
6.366.2.1	FloatBuffer	1988
6.366.2.2	~FloatBuffer	1988
6.366.3	Member Function Documentation	1988
6.366.3.1	allocate	1988
6.366.3.2	array	1989
6.366.3.3	arrayOffset	1989
6.366.3.4	asReadOnlyBuffer	1990
6.366.3.5	compact	1990

6.366.3.6	compareTo	1991
6.366.3.7	duplicate	1991
6.366.3.8	equals	1991
6.366.3.9	get	1991
6.366.3.10	get	1991
6.366.3.11	lget	1992
6.366.3.12	get	1993
6.366.3.13	hasArray	1993
6.366.3.14	operator<	1993
6.366.3.15	operator==	1993
6.366.3.16	put	1993
6.366.3.17	put	1994
6.366.3.18	put	1994
6.366.3.19	put	1995
6.366.3.20	put	1996
6.366.3.21	slice	1996
6.366.3.22	toString	1997
6.366.3.23	wrap	1997
6.366.3.24	wrap	1997
6.367	decaf::io::Flushable Class Reference	1998
6.367.1	Detailed Description	1998
6.367.2	Constructor & Destructor Documentation	1998
6.367.2.1	~Flushable	1998
6.367.3	Member Function Documentation	1998
6.367.3.1	flush	1998
6.368	activemq::commands::FlushCommand Class Reference	1999
6.368.1	Constructor & Destructor Documentation	2000
6.368.1.1	FlushCommand	2000
6.368.1.2	~FlushCommand	2000
6.368.2	Member Function Documentation	2000
6.368.2.1	cloneDataStructure	2000
6.368.2.2	copyDataStructure	2000
6.368.2.3	equals	2000
6.368.2.4	getDataStructureType	2001
6.368.2.5	toString	2001
6.368.2.6	visit	2001
6.368.3	Field Documentation	2001
6.368.3.1	ID_FLUSHCOMMAND	2001
6.369	activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller	
	Class Reference	2002
6.369.1	Detailed Description	2003
6.369.2	Constructor & Destructor Documentation	2003
6.369.2.1	FlushCommandMarshaller	2003
6.369.2.2	~FlushCommandMarshaller	2003
6.369.3	Member Function Documentation	2003
6.369.3.1	createObject	2003
6.369.3.2	getDataStructureType	2003
6.369.3.3	looseMarshal	2003
6.369.3.4	looseUnmarshal	2004
6.369.3.5	tightMarshal1	2004

6.369.3.6	tightMarshal2	2005
6.369.3.7	tightUnmarshal	2005
6.370	activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	
	Class Reference	2006
6.370.1	Detailed Description	2007
6.370.2	Constructor & Destructor Documentation	2007
6.370.2.1	FlushCommandMarshaller	2007
6.370.2.2	~FlushCommandMarshaller	2007
6.370.3	Member Function Documentation	2007
6.370.3.1	createObject	2007
6.370.3.2	getDataStructureType	2007
6.370.3.3	looseMarshal	2008
6.370.3.4	looseUnmarshal	2008
6.370.3.5	tightMarshal1	2009
6.370.3.6	tightMarshal2	2009
6.370.3.7	tightUnmarshal	2010
6.371	activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	
	Class Reference	2010
6.371.1	Detailed Description	2011
6.371.2	Constructor & Destructor Documentation	2011
6.371.2.1	FlushCommandMarshaller	2011
6.371.2.2	~FlushCommandMarshaller	2011
6.371.3	Member Function Documentation	2011
6.371.3.1	createObject	2011
6.371.3.2	getDataStructureType	2012
6.371.3.3	looseMarshal	2012
6.371.3.4	looseUnmarshal	2012
6.371.3.5	tightMarshal1	2013
6.371.3.6	tightMarshal2	2013
6.371.3.7	tightUnmarshal	2014
6.372	activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller	
	Class Reference	2014
6.372.1	Detailed Description	2015
6.372.2	Constructor & Destructor Documentation	2016
6.372.2.1	FlushCommandMarshaller	2016
6.372.2.2	~FlushCommandMarshaller	2016
6.372.3	Member Function Documentation	2016
6.372.3.1	createObject	2016
6.372.3.2	getDataStructureType	2016
6.372.3.3	looseMarshal	2016
6.372.3.4	looseUnmarshal	2017
6.372.3.5	tightMarshal1	2017
6.372.3.6	tightMarshal2	2018
6.372.3.7	tightUnmarshal	2018
6.373	activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	
	Class Reference	2019
6.373.1	Detailed Description	2020
6.373.2	Constructor & Destructor Documentation	2020
6.373.2.1	FlushCommandMarshaller	2020
6.373.2.2	~FlushCommandMarshaller	2020

6.373.3 Member Function Documentation . . . . .	2020
6.373.3.1 createObject . . . . .	2020
6.373.3.2 getDataStructureType . . . . .	2020
6.373.3.3 looseMarshal . . . . .	2020
6.373.3.4 looseUnmarshal . . . . .	2021
6.373.3.5 tightMarshal1 . . . . .	2021
6.373.3.6 tightMarshal2 . . . . .	2022
6.373.3.7 tightUnmarshal . . . . .	2022
6.374activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller	
Class Reference . . . . .	2023
6.374.1 Detailed Description . . . . .	2024
6.374.2 Constructor & Destructor Documentation . . . . .	2024
6.374.2.1 FlushCommandMarshaller . . . . .	2024
6.374.2.2 ~FlushCommandMarshaller . . . . .	2024
6.374.3 Member Function Documentation . . . . .	2024
6.374.3.1 createObject . . . . .	2024
6.374.3.2 getDataStructureType . . . . .	2024
6.374.3.3 looseMarshal . . . . .	2025
6.374.3.4 looseUnmarshal . . . . .	2025
6.374.3.5 tightMarshal1 . . . . .	2026
6.374.3.6 tightMarshal2 . . . . .	2026
6.374.3.7 tightUnmarshal . . . . .	2027
6.375decaf::util::logging::Formatter Class Reference . . . . .	2027
6.375.1 Detailed Description . . . . .	2028
6.375.2 Constructor & Destructor Documentation . . . . .	2028
6.375.2.1 ~Formatter . . . . .	2028
6.375.3 Member Function Documentation . . . . .	2028
6.375.3.1 format . . . . .	2028
6.375.3.2 formatMessage . . . . .	2028
6.375.3.3 getHead . . . . .	2029
6.375.3.4 getTail . . . . .	2029
6.376decaf::util::concurrent::Future< V > Class Template Reference . . . . .	2029
6.376.1 Detailed Description . . . . .	2030
6.376.2 Constructor & Destructor Documentation . . . . .	2030
6.376.2.1 ~Future . . . . .	2030
6.376.3 Member Function Documentation . . . . .	2030
6.376.3.1 cancel . . . . .	2030
6.376.3.2 get . . . . .	2031
6.376.3.3 get . . . . .	2031
6.376.3.4 isCancelled . . . . .	2032
6.376.3.5 isDone . . . . .	2032
6.377activemq::transport::correlator::FutureResponse Class Reference . . . . .	2032
6.377.1 Detailed Description . . . . .	2033
6.377.2 Constructor & Destructor Documentation . . . . .	2033
6.377.2.1 FutureResponse . . . . .	2033
6.377.2.2 ~FutureResponse . . . . .	2033
6.377.3 Member Function Documentation . . . . .	2033
6.377.3.1 getResponse . . . . .	2033
6.377.3.2 getResponse . . . . .	2034
6.377.3.3 getResponse . . . . .	2034

6.377.3.4	getResponse	2034
6.377.3.5	setResponse	2034
6.378	decaf::security::GeneralSecurityException Class Reference	2034
6.378.1	Constructor & Destructor Documentation	2035
6.378.1.1	GeneralSecurityException	2035
6.378.1.2	GeneralSecurityException	2035
6.378.1.3	GeneralSecurityException	2035
6.378.1.4	GeneralSecurityException	2036
6.378.1.5	GeneralSecurityException	2036
6.378.1.6	GeneralSecurityException	2036
6.378.1.7	~GeneralSecurityException	2037
6.378.2	Member Function Documentation	2037
6.378.2.1	clone	2037
6.379	decaf::internal::util::GenericResource< T > Class Template Reference	2037
6.379.1	Detailed Description	2038
6.379.2	Constructor & Destructor Documentation	2038
6.379.2.1	GenericResource	2038
6.379.2.2	~GenericResource	2038
6.379.3	Member Function Documentation	2038
6.379.3.1	getManaged	2038
6.379.3.2	setManaged	2038
6.380	gz_header_s Struct Reference	2038
6.380.1	Field Documentation	2039
6.380.1.1	comm_max	2039
6.380.1.2	comment	2039
6.380.1.3	done	2039
6.380.1.4	extra	2039
6.380.1.5	extra_len	2039
6.380.1.6	extra_max	2039
6.380.1.7	hcrc	2039
6.380.1.8	name	2039
6.380.1.9	name_max	2039
6.380.1.10	bs	2039
6.380.1.11	ltext	2039
6.380.1.12	time	2039
6.380.1.13	xflags	2039
6.381	gz_state Struct Reference	2039
6.381.1	Field Documentation	2041
6.381.1.1	direct	2041
6.381.1.2	eof	2041
6.381.1.3	err	2041
6.381.1.4	fd	2041
6.381.1.5	have	2041
6.381.1.6	how	2041
6.381.1.7	in	2041
6.381.1.8	level	2041
6.381.1.9	mode	2041
6.381.1.10	msg	2041
6.381.1.11	lnext	2041
6.381.1.12	out	2041

6.381.1.13	path . . . . .	2041
6.381.1.14	pos . . . . .	2041
6.381.1.15	raw . . . . .	2041
6.381.1.16	seek . . . . .	2041
6.381.1.17	size . . . . .	2041
6.381.1.18	skip . . . . .	2041
6.381.1.19	start . . . . .	2041
6.381.1.20	strategy . . . . .	2041
6.381.1.21	strm . . . . .	2041
6.381.1.22	want . . . . .	2041
6.382	decaf::util::logging::Handler Class Reference . . . . .	2042
6.382.1	Detailed Description . . . . .	2043
6.382.2	Constructor & Destructor Documentation . . . . .	2043
6.382.2.1	Handler . . . . .	2043
6.382.2.2	~Handler . . . . .	2043
6.382.3	Member Function Documentation . . . . .	2043
6.382.3.1	flush . . . . .	2043
6.382.3.2	getEventManager . . . . .	2043
6.382.3.3	getFilter . . . . .	2043
6.382.3.4	getFormatter . . . . .	2044
6.382.3.5	getLevel . . . . .	2044
6.382.3.6	isLoggable . . . . .	2044
6.382.3.7	publish . . . . .	2044
6.382.3.8	reportError . . . . .	2045
6.382.3.9	setEventManager . . . . .	2045
6.382.3.10	setFilter . . . . .	2045
6.382.3.11	setFormatter . . . . .	2045
6.382.3.12	setLevel . . . . .	2046
6.383	decaf::internal::util::HexStringParser Class Reference . . . . .	2046
6.383.1	Constructor & Destructor Documentation . . . . .	2047
6.383.1.1	HexStringParser . . . . .	2047
6.383.1.2	~HexStringParser . . . . .	2047
6.383.2	Member Function Documentation . . . . .	2047
6.383.2.1	parse . . . . .	2047
6.383.2.2	parseDouble . . . . .	2047
6.383.2.3	parseFloat . . . . .	2047
6.384	activemq::wireformat::openwire::utils::HexTable Class Reference . . . . .	2048
6.384.1	Detailed Description . . . . .	2048
6.384.2	Constructor & Destructor Documentation . . . . .	2048
6.384.2.1	HexTable . . . . .	2048
6.384.2.2	~HexTable . . . . .	2048
6.384.3	Member Function Documentation . . . . .	2048
6.384.3.1	operator[] . . . . .	2048
6.384.3.2	operator[] . . . . .	2049
6.384.3.3	size . . . . .	2049
6.385	decaf::net::HttpRetryException Class Reference . . . . .	2049
6.385.1	Constructor & Destructor Documentation . . . . .	2050
6.385.1.1	HttpRetryException . . . . .	2050
6.385.1.2	HttpRetryException . . . . .	2050
6.385.1.3	HttpRetryException . . . . .	2050

6.385.1.4	HttpRetryException . . . . .	2050
6.385.1.5	HttpRetryException . . . . .	2051
6.385.1.6	HttpRetryException . . . . .	2051
6.385.1.7	~HttpRetryException . . . . .	2051
6.385.2	Member Function Documentation . . . . .	2051
6.385.2.1	clone . . . . .	2051
6.386	activemq::util::IdGenerator Class Reference . . . . .	2052
6.386.1	Constructor & Destructor Documentation . . . . .	2053
6.386.1.1	IdGenerator . . . . .	2053
6.386.1.2	IdGenerator . . . . .	2053
6.386.1.3	~IdGenerator . . . . .	2053
6.386.2	Member Function Documentation . . . . .	2053
6.386.2.1	compare . . . . .	2053
6.386.2.2	generateId . . . . .	2053
6.386.2.3	getHostname . . . . .	2053
6.386.2.4	getSeedFromId . . . . .	2053
6.386.2.5	getSequenceFromId . . . . .	2054
6.387	decaf::lang::exceptions::IllegalArgumentException Class Reference . . . . .	2054
6.387.1	Constructor & Destructor Documentation . . . . .	2055
6.387.1.1	IllegalArgumentException . . . . .	2055
6.387.1.2	IllegalArgumentException . . . . .	2055
6.387.1.3	IllegalArgumentException . . . . .	2055
6.387.1.4	IllegalArgumentException . . . . .	2055
6.387.1.5	IllegalArgumentException . . . . .	2055
6.387.1.6	IllegalArgumentException . . . . .	2056
6.387.1.7	~IllegalArgumentException . . . . .	2056
6.387.2	Member Function Documentation . . . . .	2056
6.387.2.1	clone . . . . .	2056
6.388	decaf::lang::exceptions::IllegalMonitorStateException Class Reference . . . . .	2056
6.388.1	Constructor & Destructor Documentation . . . . .	2057
6.388.1.1	IllegalMonitorStateException . . . . .	2057
6.388.1.2	IllegalMonitorStateException . . . . .	2057
6.388.1.3	IllegalMonitorStateException . . . . .	2058
6.388.1.4	IllegalMonitorStateException . . . . .	2058
6.388.1.5	IllegalMonitorStateException . . . . .	2058
6.388.1.6	IllegalMonitorStateException . . . . .	2058
6.388.1.7	~IllegalMonitorStateException . . . . .	2059
6.388.2	Member Function Documentation . . . . .	2059
6.388.2.1	clone . . . . .	2059
6.389	cms::IllegalStateException Class Reference . . . . .	2059
6.389.1	Detailed Description . . . . .	2059
6.389.2	Constructor & Destructor Documentation . . . . .	2060
6.389.2.1	IllegalStateException . . . . .	2060
6.389.2.2	IllegalStateException . . . . .	2060
6.389.2.3	IllegalStateException . . . . .	2060
6.389.2.4	IllegalStateException . . . . .	2060
6.389.2.5	~IllegalStateException . . . . .	2060
6.390	decaf::lang::exceptions::IllegalStateException Class Reference . . . . .	2060
6.390.1	Constructor & Destructor Documentation . . . . .	2061
6.390.1.1	IllegalStateException . . . . .	2061

6.390.1.2	IllegalStateException	2061
6.390.1.3	IllegalStateException	2061
6.390.1.4	IllegalStateException	2061
6.390.1.5	IllegalStateException	2062
6.390.1.6	IllegalStateException	2062
6.390.1.7	~IllegalStateException	2062
6.390.2	Member Function Documentation	2062
6.390.2.1	clone	2062
6.391	decaf::lang::exceptions::IllegalThreadStateException Class Reference	2063
6.391.1	Constructor & Destructor Documentation	2064
6.391.1.1	IllegalThreadStateException	2064
6.391.1.2	IllegalThreadStateException	2064
6.391.1.3	IllegalThreadStateException	2064
6.391.1.4	IllegalThreadStateException	2064
6.391.1.5	IllegalThreadStateException	2064
6.391.1.6	IllegalThreadStateException	2065
6.391.1.7	~IllegalThreadStateException	2065
6.391.2	Member Function Documentation	2065
6.391.2.1	clone	2065
6.392	activemq::transport::inactivity::InactivityMonitor Class Reference	2065
6.392.1	Constructor & Destructor Documentation	2067
6.392.1.1	InactivityMonitor	2067
6.392.1.2	InactivityMonitor	2067
6.392.1.3	~InactivityMonitor	2067
6.392.2	Member Function Documentation	2067
6.392.2.1	close	2067
6.392.2.2	getInitialDelayTime	2067
6.392.2.3	getReadCheckTime	2067
6.392.2.4	getWriteCheckTime	2067
6.392.2.5	isKeepAliveResponseRequired	2067
6.392.2.6	onCommand	2067
6.392.2.7	oneway	2068
6.392.2.8	onException	2068
6.392.2.9	setInitialDelayTime	2069
6.392.2.10	setKeepAliveResponseRequired	2069
6.392.2.11	setReadCheckTime	2069
6.392.2.12	setWriteCheckTime	2069
6.392.3	Friends And Related Function Documentation	2069
6.392.3.1	AsyncSignalReadErrorTask	2069
6.392.3.2	AsyncWriteTask	2069
6.392.3.3	ReadChecker	2069
6.392.3.4	WriteChecker	2069
6.393	decaf::lang::exceptions::IndexOutOfBoundsException Class Reference	2069
6.393.1	Constructor & Destructor Documentation	2070
6.393.1.1	IndexOutOfBoundsException	2070
6.393.1.2	IndexOutOfBoundsException	2070
6.393.1.3	IndexOutOfBoundsException	2070
6.393.1.4	IndexOutOfBoundsException	2071
6.393.1.5	IndexOutOfBoundsException	2071
6.393.1.6	IndexOutOfBoundsException	2071



6.393.1.7 ~IndexOutOfBoundsException . . . . .	2072
6.393.2 Member Function Documentation . . . . .	2072
6.393.2.1 clone . . . . .	2072
6.394decaf::net::Inet4Address Class Reference . . . . .	2072
6.394.1 Constructor & Destructor Documentation . . . . .	2073
6.394.1.1 Inet4Address . . . . .	2073
6.394.1.2 Inet4Address . . . . .	2073
6.394.1.3 Inet4Address . . . . .	2073
6.394.1.4 ~Inet4Address . . . . .	2073
6.394.2 Member Function Documentation . . . . .	2073
6.394.2.1 isAnyLocalAddress . . . . .	2073
6.394.2.2 isLinkLocalAddress . . . . .	2074
6.394.2.3 isLoopbackAddress . . . . .	2074
6.394.2.4 isMCGlobal . . . . .	2074
6.394.2.5 isMCLinkLocal . . . . .	2074
6.394.2.6 isMCNodeLocal . . . . .	2075
6.394.2.7 isMCOrgLocal . . . . .	2075
6.394.2.8 isMCSiteLocal . . . . .	2075
6.394.2.9 isMulticastAddress . . . . .	2075
6.394.2.10 isSiteLocalAddress . . . . .	2075
6.394.3 Friends And Related Function Documentation . . . . .	2076
6.394.3.1 InetAddress . . . . .	2076
6.395decaf::net::Inet6Address Class Reference . . . . .	2076
6.395.1 Constructor & Destructor Documentation . . . . .	2077
6.395.1.1 Inet6Address . . . . .	2077
6.395.1.2 Inet6Address . . . . .	2077
6.395.1.3 Inet6Address . . . . .	2077
6.395.1.4 ~Inet6Address . . . . .	2077
6.395.2 Friends And Related Function Documentation . . . . .	2077
6.395.2.1 InetAddress . . . . .	2077
6.396decaf::net::InetAddress Class Reference . . . . .	2077
6.396.1 Detailed Description . . . . .	2079
6.396.2 Constructor & Destructor Documentation . . . . .	2080
6.396.2.1 InetAddress . . . . .	2080
6.396.2.2 InetAddress . . . . .	2080
6.396.2.3 InetAddress . . . . .	2080
6.396.2.4 ~InetAddress . . . . .	2080
6.396.3 Member Function Documentation . . . . .	2080
6.396.3.1 bytesToInt . . . . .	2080
6.396.3.2 getAddress . . . . .	2080
6.396.3.3 getAnyAddress . . . . .	2080
6.396.3.4 getByAddress . . . . .	2080
6.396.3.5 getByAddress . . . . .	2081
6.396.3.6 getHostAddress . . . . .	2081
6.396.3.7 getHostName . . . . .	2081
6.396.3.8 getLocalHost . . . . .	2082
6.396.3.9 getLoopbackAddress . . . . .	2082
6.396.3.10 isAnyLocalAddress . . . . .	2082
6.396.3.11 isLinkLocalAddress . . . . .	2082
6.396.3.12 isLoopbackAddress . . . . .	2083

6.396.3.13	isMCGlobal . . . . .	2083
6.396.3.14	isMCLinkLocal . . . . .	2083
6.396.3.15	isMCNodeLocal . . . . .	2083
6.396.3.16	isMCOrgLocal . . . . .	2083
6.396.3.17	isMCSiteLocal . . . . .	2084
6.396.3.18	isMulticastAddress . . . . .	2084
6.396.3.19	isSiteLocalAddress . . . . .	2084
6.396.3.20	toString . . . . .	2084
6.396.4	Field Documentation . . . . .	2085
6.396.4.1	addressBytes . . . . .	2085
6.396.4.2	anyBytes . . . . .	2085
6.396.4.3	hostname . . . . .	2085
6.396.4.4	loopbackBytes . . . . .	2085
6.396.4.5	reached . . . . .	2085
6.397	decaf::net::InetSocketAddress Class Reference . . . . .	2085
6.397.1	Constructor & Destructor Documentation . . . . .	2085
6.397.1.1	InetSocketAddress . . . . .	2085
6.397.1.2	~InetSocketAddress . . . . .	2085
6.398	inflate_state Struct Reference . . . . .	2085
6.398.1	Field Documentation . . . . .	2087
6.398.1.1	back . . . . .	2087
6.398.1.2	bits . . . . .	2087
6.398.1.3	check . . . . .	2087
6.398.1.4	codes . . . . .	2087
6.398.1.5	distbits . . . . .	2087
6.398.1.6	distcode . . . . .	2087
6.398.1.7	dmax . . . . .	2087
6.398.1.8	extra . . . . .	2087
6.398.1.9	flags . . . . .	2087
6.398.1.10	have . . . . .	2087
6.398.1.11	havedict . . . . .	2087
6.398.1.12	head . . . . .	2087
6.398.1.13	hold . . . . .	2087
6.398.1.14	last . . . . .	2087
6.398.1.15	lenbits . . . . .	2087
6.398.1.16	encode . . . . .	2087
6.398.1.17	length . . . . .	2087
6.398.1.18	lens . . . . .	2087
6.398.1.19	mode . . . . .	2087
6.398.1.20	ncode . . . . .	2087
6.398.1.21	ndist . . . . .	2087
6.398.1.22	next . . . . .	2087
6.398.1.23	nlen . . . . .	2087
6.398.1.24	offset . . . . .	2087
6.398.1.25	sane . . . . .	2087
6.398.1.26	total . . . . .	2087
6.398.1.27	was . . . . .	2087
6.398.1.28	wbits . . . . .	2087
6.398.1.29	whave . . . . .	2087
6.398.1.30	window . . . . .	2087

6.398.1.3	lwnext . . . . .	2087
6.398.1.32	work . . . . .	2087
6.398.1.33	wrap . . . . .	2087
6.398.1.34	wsize . . . . .	2087
6.399	decaf::util::zip::Inflater Class Reference . . . . .	2088
6.399.1	Detailed Description . . . . .	2089
6.399.2	Constructor & Destructor Documentation . . . . .	2090
6.399.2.1	Inflater . . . . .	2090
6.399.2.2	Inflater . . . . .	2090
6.399.2.3	~Inflater . . . . .	2090
6.399.3	Member Function Documentation . . . . .	2090
6.399.3.1	end . . . . .	2090
6.399.3.2	finish . . . . .	2090
6.399.3.3	finished . . . . .	2091
6.399.3.4	getAdler . . . . .	2091
6.399.3.5	getBytesRead . . . . .	2091
6.399.3.6	getBytesWritten . . . . .	2091
6.399.3.7	getRemaining . . . . .	2091
6.399.3.8	inflate . . . . .	2092
6.399.3.9	inflate . . . . .	2092
6.399.3.10	inflate . . . . .	2093
6.399.3.11	needsDictionary . . . . .	2093
6.399.3.12	needsInput . . . . .	2093
6.399.3.13	reset . . . . .	2094
6.399.3.14	setDictionary . . . . .	2094
6.399.3.15	setDictionary . . . . .	2094
6.399.3.16	setDictionary . . . . .	2095
6.399.3.17	setInput . . . . .	2095
6.399.3.18	setInput . . . . .	2096
6.399.3.19	setInput . . . . .	2096
6.400	decaf::util::zip::InflaterInputStream Class Reference . . . . .	2097
6.400.1	Detailed Description . . . . .	2100
6.400.2	Constructor & Destructor Documentation . . . . .	2100
6.400.2.1	InflaterInputStream . . . . .	2100
6.400.2.2	InflaterInputStream . . . . .	2100
6.400.2.3	InflaterInputStream . . . . .	2101
6.400.2.4	~InflaterInputStream . . . . .	2101
6.400.3	Member Function Documentation . . . . .	2101
6.400.3.1	available . . . . .	2101
6.400.3.2	close . . . . .	2102
6.400.3.3	doReadArrayBounded . . . . .	2102
6.400.3.4	doReadByte . . . . .	2102
6.400.3.5	fill . . . . .	2102
6.400.3.6	mark . . . . .	2102
6.400.3.7	markSupported . . . . .	2103
6.400.3.8	reset . . . . .	2103
6.400.3.9	skip . . . . .	2104
6.400.4	Field Documentation . . . . .	2104
6.400.4.1	atEOF . . . . .	2104
6.400.4.2	buff . . . . .	2104

6.400.4.3	DEFAULT_BUFFER_SIZE . . . . .	2105
6.400.4.4	inflater . . . . .	2105
6.400.4.5	length . . . . .	2105
6.400.4.6	ownInflater . . . . .	2105
6.401	decaf::io::InputStream Class Reference . . . . .	2105
6.401.1	Detailed Description . . . . .	2107
6.401.2	Constructor & Destructor Documentation . . . . .	2107
6.401.2.1	InputStream . . . . .	2107
6.401.2.2	~InputStream . . . . .	2107
6.401.3	Member Function Documentation . . . . .	2107
6.401.3.1	available . . . . .	2107
6.401.3.2	close . . . . .	2108
6.401.3.3	doReadArray . . . . .	2108
6.401.3.4	doReadArrayBounded . . . . .	2108
6.401.3.5	doReadByte . . . . .	2109
6.401.3.6	lock . . . . .	2109
6.401.3.7	mark . . . . .	2109
6.401.3.8	markSupported . . . . .	2109
6.401.3.9	notify . . . . .	2110
6.401.3.10	notifyAll . . . . .	2110
6.401.3.11	read . . . . .	2111
6.401.3.12	read . . . . .	2112
6.401.3.13	read . . . . .	2112
6.401.3.14	reset . . . . .	2113
6.401.3.15	skip . . . . .	2113
6.401.3.16	toString . . . . .	2114
6.401.3.17	tryLock . . . . .	2114
6.401.3.18	unlock . . . . .	2115
6.401.3.19	wait . . . . .	2115
6.401.3.20	wait . . . . .	2115
6.401.3.21	wait . . . . .	2116
6.402	decaf::io::InputStreamReader Class Reference . . . . .	2116
6.402.1	Detailed Description . . . . .	2117
6.402.2	Constructor & Destructor Documentation . . . . .	2117
6.402.2.1	InputStreamReader . . . . .	2117
6.402.2.2	~InputStreamReader . . . . .	2118
6.402.3	Member Function Documentation . . . . .	2118
6.402.3.1	checkClosed . . . . .	2118
6.402.3.2	close . . . . .	2118
6.402.3.3	doReadArrayBounded . . . . .	2118
6.402.3.4	ready . . . . .	2118
6.403	decaf::internal::nio::IntArrayBuffer Class Reference . . . . .	2119
6.403.1	Constructor & Destructor Documentation . . . . .	2123
6.403.1.1	IntArrayBuffer . . . . .	2123
6.403.1.2	IntArrayBuffer . . . . .	2123
6.403.1.3	IntArrayBuffer . . . . .	2124
6.403.1.4	IntArrayBuffer . . . . .	2124
6.403.1.5	~IntArrayBuffer . . . . .	2125
6.403.2	Member Function Documentation . . . . .	2125
6.403.2.1	array . . . . .	2125

6.403.2.2	arrayOffset . . . . .	2125
6.403.2.3	asReadOnlyBuffer . . . . .	2126
6.403.2.4	compact . . . . .	2126
6.403.2.5	duplicate . . . . .	2127
6.403.2.6	get . . . . .	2127
6.403.2.7	get . . . . .	2127
6.403.2.8	hasArray . . . . .	2128
6.403.2.9	isReadOnly . . . . .	2128
6.403.2.10	put . . . . .	2128
6.403.2.11	put . . . . .	2129
6.403.2.12	setReadOnly . . . . .	2129
6.403.2.13	slice . . . . .	2130
6.404	decaf::nio::IntBuffer Class Reference . . . . .	2130
6.404.1	Detailed Description . . . . .	2132
6.404.2	Constructor & Destructor Documentation . . . . .	2133
6.404.2.1	IntBuffer . . . . .	2133
6.404.2.2	~IntBuffer . . . . .	2133
6.404.3	Member Function Documentation . . . . .	2133
6.404.3.1	allocate . . . . .	2133
6.404.3.2	array . . . . .	2134
6.404.3.3	arrayOffset . . . . .	2134
6.404.3.4	asReadOnlyBuffer . . . . .	2135
6.404.3.5	compact . . . . .	2135
6.404.3.6	compareTo . . . . .	2136
6.404.3.7	duplicate . . . . .	2136
6.404.3.8	equals . . . . .	2136
6.404.3.9	get . . . . .	2136
6.404.3.10	get . . . . .	2136
6.404.3.11	lget . . . . .	2137
6.404.3.12	lget . . . . .	2138
6.404.3.13	hasArray . . . . .	2138
6.404.3.14	operator< . . . . .	2138
6.404.3.15	operator== . . . . .	2138
6.404.3.16	put . . . . .	2138
6.404.3.17	put . . . . .	2139
6.404.3.18	put . . . . .	2139
6.404.3.19	put . . . . .	2140
6.404.3.20	put . . . . .	2141
6.404.3.21	slice . . . . .	2141
6.404.3.22	toString . . . . .	2142
6.404.3.23	wrap . . . . .	2142
6.404.3.24	wrap . . . . .	2142
6.405	decaf::lang::Integer Class Reference . . . . .	2143
6.405.1	Constructor & Destructor Documentation . . . . .	2146
6.405.1.1	Integer . . . . .	2146
6.405.1.2	Integer . . . . .	2146
6.405.1.3	~Integer . . . . .	2147
6.405.2	Member Function Documentation . . . . .	2147
6.405.2.1	bitCount . . . . .	2147
6.405.2.2	byteValue . . . . .	2147

6.405.2.3	compareTo	2147
6.405.2.4	compareTo	2148
6.405.2.5	decode	2148
6.405.2.6	doubleValue	2148
6.405.2.7	equals	2149
6.405.2.8	equals	2149
6.405.2.9	floatValue	2149
6.405.2.10	highestOneBit	2149
6.405.2.11	intValue	2150
6.405.2.12	longValue	2150
6.405.2.13	lowestOneBit	2150
6.405.2.14	numberOfLeadingZeros	2151
6.405.2.15	numberOfTrailingZeros	2151
6.405.2.16	operator<	2151
6.405.2.17	operator<	2152
6.405.2.18	operator==	2152
6.405.2.19	operator==	2152
6.405.2.20	parseInt	2153
6.405.2.21	parseInt	2153
6.405.2.22	reverse	2154
6.405.2.23	reverseBytes	2154
6.405.2.24	rotateLeft	2154
6.405.2.25	rotateRight	2155
6.405.2.26	shortValue	2155
6.405.2.27	signum	2155
6.405.2.28	toBinaryString	2156
6.405.2.29	toHexString	2156
6.405.2.30	toOctalString	2157
6.405.2.31	toString	2157
6.405.2.32	toString	2157
6.405.2.33	toString	2158
6.405.2.34	valueOf	2158
6.405.2.35	valueOf	2158
6.405.2.36	valueOf	2159
6.405.3	Field Documentation	2159
6.405.3.1	MAX_VALUE	2159
6.405.3.2	MIN_VALUE	2159
6.405.3.3	SIZE	2159
6.406	activemq::commands::IntegerResponse Class Reference	2160
6.406.1	Constructor & Destructor Documentation	2161
6.406.1.1	IntegerResponse	2161
6.406.1.2	~IntegerResponse	2161
6.406.2	Member Function Documentation	2161
6.406.2.1	cloneDataStructure	2161
6.406.2.2	copyDataStructure	2161
6.406.2.3	equals	2161
6.406.2.4	getDataStructureType	2162
6.406.2.5	getResult	2162
6.406.2.6	setResult	2162
6.406.2.7	toString	2162

6.406.3 Field Documentation . . . . .	2162
6.406.3.1 ID_INTEGERRESPONSE . . . . .	2162
6.406.3.2 result . . . . .	2162
6.407activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller	
Class Reference . . . . .	2162
6.407.1 Detailed Description . . . . .	2163
6.407.2 Constructor & Destructor Documentation . . . . .	2164
6.407.2.1 IntegerResponseMarshaller . . . . .	2164
6.407.2.2 ~IntegerResponseMarshaller . . . . .	2164
6.407.3 Member Function Documentation . . . . .	2164
6.407.3.1 createObject . . . . .	2164
6.407.3.2 getDataStructureType . . . . .	2164
6.407.3.3 looseMarshal . . . . .	2164
6.407.3.4 looseUnmarshal . . . . .	2165
6.407.3.5 tightMarshal1 . . . . .	2165
6.407.3.6 tightMarshal2 . . . . .	2166
6.407.3.7 tightUnmarshal . . . . .	2166
6.408activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller	
Class Reference . . . . .	2167
6.408.1 Detailed Description . . . . .	2168
6.408.2 Constructor & Destructor Documentation . . . . .	2168
6.408.2.1 IntegerResponseMarshaller . . . . .	2168
6.408.2.2 ~IntegerResponseMarshaller . . . . .	2168
6.408.3 Member Function Documentation . . . . .	2168
6.408.3.1 createObject . . . . .	2168
6.408.3.2 getDataStructureType . . . . .	2168
6.408.3.3 looseMarshal . . . . .	2169
6.408.3.4 looseUnmarshal . . . . .	2169
6.408.3.5 tightMarshal1 . . . . .	2169
6.408.3.6 tightMarshal2 . . . . .	2170
6.408.3.7 tightUnmarshal . . . . .	2170
6.409activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller	
Class Reference . . . . .	2171
6.409.1 Detailed Description . . . . .	2172
6.409.2 Constructor & Destructor Documentation . . . . .	2172
6.409.2.1 IntegerResponseMarshaller . . . . .	2172
6.409.2.2 ~IntegerResponseMarshaller . . . . .	2172
6.409.3 Member Function Documentation . . . . .	2172
6.409.3.1 createObject . . . . .	2172
6.409.3.2 getDataStructureType . . . . .	2173
6.409.3.3 looseMarshal . . . . .	2173
6.409.3.4 looseUnmarshal . . . . .	2173
6.409.3.5 tightMarshal1 . . . . .	2174
6.409.3.6 tightMarshal2 . . . . .	2174
6.409.3.7 tightUnmarshal . . . . .	2175
6.410activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller	
Class Reference . . . . .	2175
6.410.1 Detailed Description . . . . .	2176
6.410.2 Constructor & Destructor Documentation . . . . .	2177
6.410.2.1 IntegerResponseMarshaller . . . . .	2177

6.410.2.2 ~IntegerResponseMarshaller . . . . .	2177
6.410.3 Member Function Documentation . . . . .	2177
6.410.3.1 createObject . . . . .	2177
6.410.3.2 getDataStructureType . . . . .	2177
6.410.3.3 looseMarshal . . . . .	2177
6.410.3.4 looseUnmarshal . . . . .	2178
6.410.3.5 tightMarshal1 . . . . .	2178
6.410.3.6 tightMarshal2 . . . . .	2179
6.410.3.7 tightUnmarshal . . . . .	2179
6.411activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller	
Class Reference . . . . .	2180
6.411.1 Detailed Description . . . . .	2181
6.411.2 Constructor & Destructor Documentation . . . . .	2181
6.411.2.1 IntegerResponseMarshaller . . . . .	2181
6.411.2.2 ~IntegerResponseMarshaller . . . . .	2181
6.411.3 Member Function Documentation . . . . .	2181
6.411.3.1 createObject . . . . .	2181
6.411.3.2 getDataStructureType . . . . .	2181
6.411.3.3 looseMarshal . . . . .	2182
6.411.3.4 looseUnmarshal . . . . .	2182
6.411.3.5 tightMarshal1 . . . . .	2182
6.411.3.6 tightMarshal2 . . . . .	2183
6.411.3.7 tightUnmarshal . . . . .	2183
6.412activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller	
Class Reference . . . . .	2184
6.412.1 Detailed Description . . . . .	2185
6.412.2 Constructor & Destructor Documentation . . . . .	2185
6.412.2.1 IntegerResponseMarshaller . . . . .	2185
6.412.2.2 ~IntegerResponseMarshaller . . . . .	2185
6.412.3 Member Function Documentation . . . . .	2185
6.412.3.1 createObject . . . . .	2185
6.412.3.2 getDataStructureType . . . . .	2186
6.412.3.3 looseMarshal . . . . .	2186
6.412.3.4 looseUnmarshal . . . . .	2186
6.412.3.5 tightMarshal1 . . . . .	2187
6.412.3.6 tightMarshal2 . . . . .	2187
6.412.3.7 tightUnmarshal . . . . .	2188
6.413internal_state Struct Reference . . . . .	2188
6.413.1 Field Documentation . . . . .	2191
6.413.1.1 bi_buf . . . . .	2191
6.413.1.2 bi_valid . . . . .	2191
6.413.1.3 bl_count . . . . .	2191
6.413.1.4 bl_desc . . . . .	2191
6.413.1.5 bl_tree . . . . .	2191
6.413.1.6 block_start . . . . .	2191
6.413.1.7 d_buf . . . . .	2191
6.413.1.8 d_desc . . . . .	2191
6.413.1.9 depth . . . . .	2191
6.413.1.10dummy . . . . .	2191
6.413.1.1ldyn_dtree . . . . .	2191



6.413.1.12	dyn_ltree . . . . .	2191
6.413.1.13	good_match . . . . .	2191
6.413.1.14	gzhead . . . . .	2191
6.413.1.15	gzindex . . . . .	2191
6.413.1.16	hash_bits . . . . .	2191
6.413.1.17	hash_mask . . . . .	2191
6.413.1.18	hash_shift . . . . .	2191
6.413.1.19	hash_size . . . . .	2191
6.413.1.20	head . . . . .	2191
6.413.1.21	heap . . . . .	2191
6.413.1.22	heap_len . . . . .	2191
6.413.1.23	heap_max . . . . .	2191
6.413.1.24	high_water . . . . .	2191
6.413.1.25	ins_h . . . . .	2191
6.413.1.26	lit_buf . . . . .	2191
6.413.1.27	lit_desc . . . . .	2191
6.413.1.28	last_eob_len . . . . .	2191
6.413.1.29	last_flush . . . . .	2191
6.413.1.30	last_lit . . . . .	2191
6.413.1.31	llevel . . . . .	2191
6.413.1.32	lit_bufsize . . . . .	2191
6.413.1.33	lookahead . . . . .	2191
6.413.1.34	match_available . . . . .	2191
6.413.1.35	match_length . . . . .	2191
6.413.1.36	match_start . . . . .	2191
6.413.1.37	matches . . . . .	2191
6.413.1.38	max_chain_length . . . . .	2191
6.413.1.39	max_lazy_match . . . . .	2191
6.413.1.40	method . . . . .	2191
6.413.1.41	nice_match . . . . .	2191
6.413.1.42	opt_len . . . . .	2191
6.413.1.43	pending . . . . .	2191
6.413.1.44	pending_buf . . . . .	2191
6.413.1.45	pending_buf_size . . . . .	2191
6.413.1.46	pending_out . . . . .	2191
6.413.1.47	prev . . . . .	2191
6.413.1.48	prev_length . . . . .	2191
6.413.1.49	prev_match . . . . .	2191
6.413.1.50	static_len . . . . .	2191
6.413.1.51	status . . . . .	2191
6.413.1.52	strategy . . . . .	2191
6.413.1.53	strm . . . . .	2191
6.413.1.54	strstart . . . . .	2191
6.413.1.55	w_bits . . . . .	2191
6.413.1.56	w_mask . . . . .	2191
6.413.1.57	w_size . . . . .	2191
6.413.1.58	window . . . . .	2191
6.413.1.59	window_size . . . . .	2191
6.413.1.60	wrap . . . . .	2191
6.414	activemq::transport::mock::InternalCommandListener Class Reference	2192

6.414.1 Detailed Description . . . . .	2192
6.414.2 Constructor & Destructor Documentation . . . . .	2193
6.414.2.1 InternalCommandListener . . . . .	2193
6.414.2.2 ~InternalCommandListener . . . . .	2193
6.414.3 Member Function Documentation . . . . .	2193
6.414.3.1 onCommand . . . . .	2193
6.414.3.2 run . . . . .	2193
6.414.3.3 setResponseBuilder . . . . .	2193
6.414.3.4 setTransport . . . . .	2193
6.415decaf::lang::exceptions::InterruptedException Class Reference . . . . .	2193
6.415.1 Constructor & Destructor Documentation . . . . .	2194
6.415.1.1 InterruptedException . . . . .	2194
6.415.1.2 InterruptedException . . . . .	2194
6.415.1.3 InterruptedException . . . . .	2195
6.415.1.4 InterruptedException . . . . .	2195
6.415.1.5 InterruptedException . . . . .	2195
6.415.1.6 InterruptedException . . . . .	2195
6.415.1.7 ~InterruptedException . . . . .	2196
6.415.2 Member Function Documentation . . . . .	2196
6.415.2.1 clone . . . . .	2196
6.416decaf::io::InterruptedException Class Reference . . . . .	2196
6.416.1 Constructor & Destructor Documentation . . . . .	2197
6.416.1.1 InterruptedException . . . . .	2197
6.416.1.2 InterruptedException . . . . .	2197
6.416.1.3 InterruptedException . . . . .	2197
6.416.1.4 InterruptedException . . . . .	2197
6.416.1.5 InterruptedException . . . . .	2198
6.416.1.6 InterruptedException . . . . .	2198
6.416.1.7 ~InterruptedException . . . . .	2198
6.416.2 Member Function Documentation . . . . .	2198
6.416.2.1 clone . . . . .	2198
6.417cms::InvalidClientIdException Class Reference . . . . .	2199
6.417.1 Detailed Description . . . . .	2199
6.417.2 Constructor & Destructor Documentation . . . . .	2200
6.417.2.1 InvalidClientIdException . . . . .	2200
6.417.2.2 InvalidClientIdException . . . . .	2200
6.417.2.3 InvalidClientIdException . . . . .	2200
6.417.2.4 InvalidClientIdException . . . . .	2200
6.417.2.5 ~InvalidClientIdException . . . . .	2200
6.418cms::InvalidDestinationException Class Reference . . . . .	2200
6.418.1 Detailed Description . . . . .	2200
6.418.2 Constructor & Destructor Documentation . . . . .	2201
6.418.2.1 InvalidDestinationException . . . . .	2201
6.418.2.2 InvalidDestinationException . . . . .	2201
6.418.2.3 InvalidDestinationException . . . . .	2201
6.418.2.4 InvalidDestinationException . . . . .	2201
6.418.2.5 ~InvalidDestinationException . . . . .	2201
6.419decaf::security::InvalidKeyException Class Reference . . . . .	2201
6.419.1 Constructor & Destructor Documentation . . . . .	2202
6.419.1.1 InvalidKeyException . . . . .	2202

6.419.1.2 InvalidKeyException . . . . .	2202
6.419.1.3 InvalidKeyException . . . . .	2202
6.419.1.4 InvalidKeyException . . . . .	2202
6.419.1.5 InvalidKeyException . . . . .	2203
6.419.1.6 InvalidKeyException . . . . .	2203
6.419.1.7 ~InvalidKeyException . . . . .	2203
6.419.2 Member Function Documentation . . . . .	2203
6.419.2.1 clone . . . . .	2203
6.420decaf::nio::InvalidMarkException Class Reference . . . . .	2204
6.420.1 Constructor & Destructor Documentation . . . . .	2204
6.420.1.1 InvalidMarkException . . . . .	2204
6.420.1.2 InvalidMarkException . . . . .	2205
6.420.1.3 InvalidMarkException . . . . .	2205
6.420.1.4 InvalidMarkException . . . . .	2205
6.420.1.5 InvalidMarkException . . . . .	2205
6.420.1.6 InvalidMarkException . . . . .	2206
6.420.1.7 ~InvalidMarkException . . . . .	2206
6.420.2 Member Function Documentation . . . . .	2206
6.420.2.1 clone . . . . .	2206
6.421cms::InvalidSelectorException Class Reference . . . . .	2206
6.421.1 Detailed Description . . . . .	2207
6.421.2 Constructor & Destructor Documentation . . . . .	2207
6.421.2.1 InvalidSelectorException . . . . .	2207
6.421.2.2 InvalidSelectorException . . . . .	2207
6.421.2.3 InvalidSelectorException . . . . .	2207
6.421.2.4 InvalidSelectorException . . . . .	2207
6.421.2.5 ~InvalidSelectorException . . . . .	2207
6.422decaf::lang::exceptions::InvalidStateException Class Reference . . . . .	2207
6.422.1 Constructor & Destructor Documentation . . . . .	2208
6.422.1.1 InvalidStateException . . . . .	2208
6.422.1.2 InvalidStateException . . . . .	2208
6.422.1.3 InvalidStateException . . . . .	2209
6.422.1.4 InvalidStateException . . . . .	2209
6.422.1.5 InvalidStateException . . . . .	2209
6.422.1.6 InvalidStateException . . . . .	2209
6.422.1.7 ~InvalidStateException . . . . .	2210
6.422.2 Member Function Documentation . . . . .	2210
6.422.2.1 clone . . . . .	2210
6.423decaf::io::IOException Class Reference . . . . .	2210
6.423.1 Constructor & Destructor Documentation . . . . .	2211
6.423.1.1 IOException . . . . .	2211
6.423.1.2 IOException . . . . .	2211
6.423.1.3 IOException . . . . .	2211
6.423.1.4 IOException . . . . .	2211
6.423.1.5 IOException . . . . .	2212
6.423.1.6 IOException . . . . .	2212
6.423.1.7 ~IOException . . . . .	2212
6.423.2 Member Function Documentation . . . . .	2212
6.423.2.1 clone . . . . .	2212
6.424activemq::transport::IOTransport Class Reference . . . . .	2213

6.424.1 Detailed Description . . . . .	2215
6.424.2 Constructor & Destructor Documentation . . . . .	2215
6.424.2.1 IOTransport . . . . .	2215
6.424.2.2 IOTransport . . . . .	2215
6.424.2.3 ~IOTransport . . . . .	2215
6.424.3 Member Function Documentation . . . . .	2215
6.424.3.1 close . . . . .	2215
6.424.3.2 getRemoteAddress . . . . .	2216
6.424.3.3 getTransportListener . . . . .	2216
6.424.3.4 isClosed . . . . .	2216
6.424.3.5 isConnected . . . . .	2216
6.424.3.6 isFaultTolerant . . . . .	2217
6.424.3.7 narrow . . . . .	2217
6.424.3.8 oneway . . . . .	2217
6.424.3.9 reconnect . . . . .	2218
6.424.3.10request . . . . .	2218
6.424.3.11request . . . . .	2218
6.424.3.12run . . . . .	2219
6.424.3.13setInputStream . . . . .	2219
6.424.3.14setOutputStream . . . . .	2219
6.424.3.15setTransportListener . . . . .	2219
6.424.3.16setWireFormat . . . . .	2220
6.424.3.17start . . . . .	2220
6.424.3.18stop . . . . .	2220
6.425decaf::lang::Iterable< E > Class Template Reference . . . . .	2220
6.425.1 Detailed Description . . . . .	2221
6.425.2 Constructor & Destructor Documentation . . . . .	2221
6.425.2.1 ~Iterable . . . . .	2221
6.425.3 Member Function Documentation . . . . .	2221
6.425.3.1 iterator . . . . .	2221
6.425.3.2 iterator . . . . .	2222
6.426decaf::util::Iterator< T > Class Template Reference . . . . .	2222
6.426.1 Detailed Description . . . . .	2223
6.426.2 Constructor & Destructor Documentation . . . . .	2223
6.426.2.1 ~Iterator . . . . .	2223
6.426.3 Member Function Documentation . . . . .	2223
6.426.3.1 hasNext . . . . .	2223
6.426.3.2 next . . . . .	2223
6.426.3.3 remove . . . . .	2223
6.427activemq::commands::JournalQueueAck Class Reference . . . . .	2224
6.427.1 Constructor & Destructor Documentation . . . . .	2225
6.427.1.1 JournalQueueAck . . . . .	2225
6.427.1.2 ~JournalQueueAck . . . . .	2225
6.427.2 Member Function Documentation . . . . .	2225
6.427.2.1 cloneDataStructure . . . . .	2225
6.427.2.2 copyDataStructure . . . . .	2225
6.427.2.3 equals . . . . .	2226
6.427.2.4 getDataStructureType . . . . .	2226
6.427.2.5 getDestination . . . . .	2227
6.427.2.6 getDestination . . . . .	2227

6.427.2.7	getMessageAck	2227
6.427.2.8	getMessageAck	2227
6.427.2.9	setDestination	2227
6.427.2.10	setMessageAck	2227
6.427.2.1	ltoString	2227
6.427.3	Field Documentation	2227
6.427.3.1	destination	2227
6.427.3.2	ID_JOURNALQUEUEACK	2227
6.427.3.3	messageAck	2227
6.428	activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller	
	Class Reference	2228
6.428.1	Detailed Description	2229
6.428.2	Constructor & Destructor Documentation	2229
6.428.2.1	JournalQueueAckMarshaller	2229
6.428.2.2	~JournalQueueAckMarshaller	2229
6.428.3	Member Function Documentation	2229
6.428.3.1	createObject	2229
6.428.3.2	getDataStructureType	2229
6.428.3.3	looseMarshal	2229
6.428.3.4	looseUnmarshal	2230
6.428.3.5	tightMarshal1	2230
6.428.3.6	tightMarshal2	2231
6.428.3.7	tightUnmarshal	2231
6.429	activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	
	Class Reference	2232
6.429.1	Detailed Description	2233
6.429.2	Constructor & Destructor Documentation	2233
6.429.2.1	JournalQueueAckMarshaller	2233
6.429.2.2	~JournalQueueAckMarshaller	2233
6.429.3	Member Function Documentation	2233
6.429.3.1	createObject	2233
6.429.3.2	getDataStructureType	2233
6.429.3.3	looseMarshal	2233
6.429.3.4	looseUnmarshal	2234
6.429.3.5	tightMarshal1	2234
6.429.3.6	tightMarshal2	2235
6.429.3.7	tightUnmarshal	2235
6.430	activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller	
	Class Reference	2236
6.430.1	Detailed Description	2237
6.430.2	Constructor & Destructor Documentation	2237
6.430.2.1	JournalQueueAckMarshaller	2237
6.430.2.2	~JournalQueueAckMarshaller	2237
6.430.3	Member Function Documentation	2237
6.430.3.1	createObject	2237
6.430.3.2	getDataStructureType	2237
6.430.3.3	looseMarshal	2237
6.430.3.4	looseUnmarshal	2238
6.430.3.5	tightMarshal1	2238
6.430.3.6	tightMarshal2	2239

6.430.3.7 tightUnmarshal . . . . .	2239
6.431activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	
Class Reference . . . . .	2240
6.431.1 Detailed Description . . . . .	2241
6.431.2 Constructor & Destructor Documentation . . . . .	2241
6.431.2.1 JournalQueueAckMarshaller . . . . .	2241
6.431.2.2 ~JournalQueueAckMarshaller . . . . .	2241
6.431.3 Member Function Documentation . . . . .	2241
6.431.3.1 createObject . . . . .	2241
6.431.3.2 getDataStructureType . . . . .	2241
6.431.3.3 looseMarshal . . . . .	2241
6.431.3.4 looseUnmarshal . . . . .	2242
6.431.3.5 tightMarshal1 . . . . .	2242
6.431.3.6 tightMarshal2 . . . . .	2243
6.431.3.7 tightUnmarshal . . . . .	2243
6.432activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	
Class Reference . . . . .	2244
6.432.1 Detailed Description . . . . .	2245
6.432.2 Constructor & Destructor Documentation . . . . .	2245
6.432.2.1 JournalQueueAckMarshaller . . . . .	2245
6.432.2.2 ~JournalQueueAckMarshaller . . . . .	2245
6.432.3 Member Function Documentation . . . . .	2245
6.432.3.1 createObject . . . . .	2245
6.432.3.2 getDataStructureType . . . . .	2245
6.432.3.3 looseMarshal . . . . .	2245
6.432.3.4 looseUnmarshal . . . . .	2246
6.432.3.5 tightMarshal1 . . . . .	2246
6.432.3.6 tightMarshal2 . . . . .	2247
6.432.3.7 tightUnmarshal . . . . .	2247
6.433activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	
Class Reference . . . . .	2248
6.433.1 Detailed Description . . . . .	2249
6.433.2 Constructor & Destructor Documentation . . . . .	2249
6.433.2.1 JournalQueueAckMarshaller . . . . .	2249
6.433.2.2 ~JournalQueueAckMarshaller . . . . .	2249
6.433.3 Member Function Documentation . . . . .	2249
6.433.3.1 createObject . . . . .	2249
6.433.3.2 getDataStructureType . . . . .	2249
6.433.3.3 looseMarshal . . . . .	2249
6.433.3.4 looseUnmarshal . . . . .	2250
6.433.3.5 tightMarshal1 . . . . .	2250
6.433.3.6 tightMarshal2 . . . . .	2251
6.433.3.7 tightUnmarshal . . . . .	2251
6.434activemq::commands::JournalTopicAck Class Reference . . . . .	2252
6.434.1 Constructor & Destructor Documentation . . . . .	2253
6.434.1.1 JournalTopicAck . . . . .	2253
6.434.1.2 ~JournalTopicAck . . . . .	2253
6.434.2 Member Function Documentation . . . . .	2253
6.434.2.1 cloneDataStructure . . . . .	2253
6.434.2.2 copyDataStructure . . . . .	2253

6.434.2.3	equals	2254
6.434.2.4	getClientId	2254
6.434.2.5	getClientId	2254
6.434.2.6	getDataStructureType	2254
6.434.2.7	getDestination	2255
6.434.2.8	getDestination	2255
6.434.2.9	getMessageId	2255
6.434.2.10	getMessageId	2255
6.434.2.11	getMessageSequenceId	2255
6.434.2.12	getSubscriptionName	2255
6.434.2.13	getSubscriptionName	2255
6.434.2.14	getTransactionId	2255
6.434.2.15	getTransactionId	2255
6.434.2.16	setClientId	2255
6.434.2.17	setDestination	2255
6.434.2.18	setMessageId	2255
6.434.2.19	setMessageSequenceId	2255
6.434.2.20	setSubscriptionName	2255
6.434.2.21	setTransactionId	2255
6.434.2.22	toString	2255
6.434.3	Field Documentation	2256
6.434.3.1	clientId	2256
6.434.3.2	destination	2256
6.434.3.3	ID_JOURNALTOPICACK	2256
6.434.3.4	messageId	2256
6.434.3.5	messageSequenceId	2256
6.434.3.6	subscriptionName	2256
6.434.3.7	transactionId	2256
6.435	activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller	
	Class Reference	2256
6.435.1	Detailed Description	2257
6.435.2	Constructor & Destructor Documentation	2258
6.435.2.1	JournalTopicAckMarshaller	2258
6.435.2.2	~JournalTopicAckMarshaller	2258
6.435.3	Member Function Documentation	2258
6.435.3.1	createObject	2258
6.435.3.2	getDataStructureType	2258
6.435.3.3	looseMarshal	2258
6.435.3.4	looseUnmarshal	2259
6.435.3.5	tightMarshal1	2259
6.435.3.6	tightMarshal2	2260
6.435.3.7	tightUnmarshal	2260
6.436	activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	
	Class Reference	2261
6.436.1	Detailed Description	2262
6.436.2	Constructor & Destructor Documentation	2262
6.436.2.1	JournalTopicAckMarshaller	2262
6.436.2.2	~JournalTopicAckMarshaller	2262
6.436.3	Member Function Documentation	2262
6.436.3.1	createObject	2262

6.436.3.2	getDataStructureType	2262
6.436.3.3	looseMarshal	2262
6.436.3.4	looseUnmarshal	2263
6.436.3.5	tightMarshal1	2263
6.436.3.6	tightMarshal2	2264
6.436.3.7	tightUnmarshal	2264
6.437	activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller	
	Class Reference	2265
6.437.1	Detailed Description	2266
6.437.2	Constructor & Destructor Documentation	2266
6.437.2.1	JournalTopicAckMarshaller	2266
6.437.2.2	~JournalTopicAckMarshaller	2266
6.437.3	Member Function Documentation	2266
6.437.3.1	createObject	2266
6.437.3.2	getDataStructureType	2266
6.437.3.3	looseMarshal	2266
6.437.3.4	looseUnmarshal	2267
6.437.3.5	tightMarshal1	2267
6.437.3.6	tightMarshal2	2268
6.437.3.7	tightUnmarshal	2268
6.438	activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller	
	Class Reference	2269
6.438.1	Detailed Description	2270
6.438.2	Constructor & Destructor Documentation	2270
6.438.2.1	JournalTopicAckMarshaller	2270
6.438.2.2	~JournalTopicAckMarshaller	2270
6.438.3	Member Function Documentation	2270
6.438.3.1	createObject	2270
6.438.3.2	getDataStructureType	2270
6.438.3.3	looseMarshal	2270
6.438.3.4	looseUnmarshal	2271
6.438.3.5	tightMarshal1	2271
6.438.3.6	tightMarshal2	2272
6.438.3.7	tightUnmarshal	2272
6.439	activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller	
	Class Reference	2273
6.439.1	Detailed Description	2274
6.439.2	Constructor & Destructor Documentation	2274
6.439.2.1	JournalTopicAckMarshaller	2274
6.439.2.2	~JournalTopicAckMarshaller	2274
6.439.3	Member Function Documentation	2274
6.439.3.1	createObject	2274
6.439.3.2	getDataStructureType	2274
6.439.3.3	looseMarshal	2274
6.439.3.4	looseUnmarshal	2275
6.439.3.5	tightMarshal1	2275
6.439.3.6	tightMarshal2	2276
6.439.3.7	tightUnmarshal	2276
6.440	activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller	
	Class Reference	2277



6.440.1 Detailed Description . . . . .	2278
6.440.2 Constructor & Destructor Documentation . . . . .	2278
6.440.2.1 JournalTopicAckMarshaller . . . . .	2278
6.440.2.2 ~JournalTopicAckMarshaller . . . . .	2278
6.440.3 Member Function Documentation . . . . .	2278
6.440.3.1 createObject . . . . .	2278
6.440.3.2 getDataStructureType . . . . .	2278
6.440.3.3 looseMarshal . . . . .	2278
6.440.3.4 looseUnmarshal . . . . .	2279
6.440.3.5 tightMarshal1 . . . . .	2279
6.440.3.6 tightMarshal2 . . . . .	2280
6.440.3.7 tightUnmarshal . . . . .	2280
6.441 activemq::commands::JournalTrace Class Reference . . . . .	2281
6.441.1 Constructor & Destructor Documentation . . . . .	2282
6.441.1.1 JournalTrace . . . . .	2282
6.441.1.2 ~JournalTrace . . . . .	2282
6.441.2 Member Function Documentation . . . . .	2282
6.441.2.1 cloneDataStructure . . . . .	2282
6.441.2.2 copyDataStructure . . . . .	2282
6.441.2.3 equals . . . . .	2282
6.441.2.4 getDataStructureType . . . . .	2283
6.441.2.5 getMessage . . . . .	2283
6.441.2.6 getMessage . . . . .	2283
6.441.2.7 setMessage . . . . .	2283
6.441.2.8 toString . . . . .	2283
6.441.3 Field Documentation . . . . .	2283
6.441.3.1 ID_JOURNALTRACE . . . . .	2283
6.441.3.2 message . . . . .	2283
6.442 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller	
Class Reference . . . . .	2283
6.442.1 Detailed Description . . . . .	2284
6.442.2 Constructor & Destructor Documentation . . . . .	2285
6.442.2.1 JournalTraceMarshaller . . . . .	2285
6.442.2.2 ~JournalTraceMarshaller . . . . .	2285
6.442.3 Member Function Documentation . . . . .	2285
6.442.3.1 createObject . . . . .	2285
6.442.3.2 getDataStructureType . . . . .	2285
6.442.3.3 looseMarshal . . . . .	2285
6.442.3.4 looseUnmarshal . . . . .	2286
6.442.3.5 tightMarshal1 . . . . .	2286
6.442.3.6 tightMarshal2 . . . . .	2287
6.442.3.7 tightUnmarshal . . . . .	2287
6.443 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller	
Class Reference . . . . .	2288
6.443.1 Detailed Description . . . . .	2289
6.443.2 Constructor & Destructor Documentation . . . . .	2289
6.443.2.1 JournalTraceMarshaller . . . . .	2289
6.443.2.2 ~JournalTraceMarshaller . . . . .	2289
6.443.3 Member Function Documentation . . . . .	2289
6.443.3.1 createObject . . . . .	2289

6.443.3.2	getDataStructureType	2289
6.443.3.3	looseMarshal	2289
6.443.3.4	looseUnmarshal	2290
6.443.3.5	tightMarshal1	2290
6.443.3.6	tightMarshal2	2291
6.443.3.7	tightUnmarshal	2291
6.444	activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller	
	Class Reference	2292
6.444.1	Detailed Description	2293
6.444.2	Constructor & Destructor Documentation	2293
6.444.2.1	JournalTraceMarshaller	2293
6.444.2.2	~JournalTraceMarshaller	2293
6.444.3	Member Function Documentation	2293
6.444.3.1	createObject	2293
6.444.3.2	getDataStructureType	2293
6.444.3.3	looseMarshal	2293
6.444.3.4	looseUnmarshal	2294
6.444.3.5	tightMarshal1	2294
6.444.3.6	tightMarshal2	2295
6.444.3.7	tightUnmarshal	2295
6.445	activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller	
	Class Reference	2296
6.445.1	Detailed Description	2297
6.445.2	Constructor & Destructor Documentation	2297
6.445.2.1	JournalTraceMarshaller	2297
6.445.2.2	~JournalTraceMarshaller	2297
6.445.3	Member Function Documentation	2297
6.445.3.1	createObject	2297
6.445.3.2	getDataStructureType	2297
6.445.3.3	looseMarshal	2297
6.445.3.4	looseUnmarshal	2298
6.445.3.5	tightMarshal1	2298
6.445.3.6	tightMarshal2	2299
6.445.3.7	tightUnmarshal	2299
6.446	activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller	
	Class Reference	2300
6.446.1	Detailed Description	2301
6.446.2	Constructor & Destructor Documentation	2301
6.446.2.1	JournalTraceMarshaller	2301
6.446.2.2	~JournalTraceMarshaller	2301
6.446.3	Member Function Documentation	2301
6.446.3.1	createObject	2301
6.446.3.2	getDataStructureType	2301
6.446.3.3	looseMarshal	2301
6.446.3.4	looseUnmarshal	2302
6.446.3.5	tightMarshal1	2302
6.446.3.6	tightMarshal2	2303
6.446.3.7	tightUnmarshal	2303
6.447	activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller	
	Class Reference	2304

6.447.1 Detailed Description . . . . .	2305
6.447.2 Constructor & Destructor Documentation . . . . .	2305
6.447.2.1 JournalTraceMarshaller . . . . .	2305
6.447.2.2 ~JournalTraceMarshaller . . . . .	2305
6.447.3 Member Function Documentation . . . . .	2305
6.447.3.1 createObject . . . . .	2305
6.447.3.2 getDataStructureType . . . . .	2305
6.447.3.3 looseMarshal . . . . .	2305
6.447.3.4 looseUnmarshal . . . . .	2306
6.447.3.5 tightMarshal1 . . . . .	2306
6.447.3.6 tightMarshal2 . . . . .	2307
6.447.3.7 tightUnmarshal . . . . .	2307
6.448activemq::commands::JournalTransaction Class Reference . . . . .	2308
6.448.1 Constructor & Destructor Documentation . . . . .	2309
6.448.1.1 JournalTransaction . . . . .	2309
6.448.1.2 ~JournalTransaction . . . . .	2309
6.448.2 Member Function Documentation . . . . .	2309
6.448.2.1 cloneDataStructure . . . . .	2309
6.448.2.2 copyDataStructure . . . . .	2309
6.448.2.3 equals . . . . .	2309
6.448.2.4 getDataStructureType . . . . .	2310
6.448.2.5 getTransactionId . . . . .	2310
6.448.2.6 getTransactionId . . . . .	2310
6.448.2.7 getType . . . . .	2310
6.448.2.8 getWasPrepared . . . . .	2310
6.448.2.9 setTransactionId . . . . .	2310
6.448.2.10setType . . . . .	2310
6.448.2.11setWasPrepared . . . . .	2310
6.448.2.12toString . . . . .	2310
6.448.3 Field Documentation . . . . .	2311
6.448.3.1 ID_JOURNALTRANSACTION . . . . .	2311
6.448.3.2 transactionId . . . . .	2311
6.448.3.3 type . . . . .	2311
6.448.3.4 wasPrepared . . . . .	2311
6.449activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller	
Class Reference . . . . .	2311
6.449.1 Detailed Description . . . . .	2312
6.449.2 Constructor & Destructor Documentation . . . . .	2312
6.449.2.1 JournalTransactionMarshaller . . . . .	2312
6.449.2.2 ~JournalTransactionMarshaller . . . . .	2312
6.449.3 Member Function Documentation . . . . .	2312
6.449.3.1 createObject . . . . .	2312
6.449.3.2 getDataStructureType . . . . .	2313
6.449.3.3 looseMarshal . . . . .	2313
6.449.3.4 looseUnmarshal . . . . .	2313
6.449.3.5 tightMarshal1 . . . . .	2314
6.449.3.6 tightMarshal2 . . . . .	2314
6.449.3.7 tightUnmarshal . . . . .	2315
6.450activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller	
Class Reference . . . . .	2315

6.450.1 Detailed Description . . . . .	2316
6.450.2 Constructor & Destructor Documentation . . . . .	2316
6.450.2.1 JournalTransactionMarshaller . . . . .	2316
6.450.2.2 ~JournalTransactionMarshaller . . . . .	2316
6.450.3 Member Function Documentation . . . . .	2316
6.450.3.1 createObject . . . . .	2316
6.450.3.2 getDataStructureType . . . . .	2317
6.450.3.3 looseMarshal . . . . .	2317
6.450.3.4 looseUnmarshal . . . . .	2317
6.450.3.5 tightMarshal1 . . . . .	2318
6.450.3.6 tightMarshal2 . . . . .	2318
6.450.3.7 tightUnmarshal . . . . .	2319
6.451activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller	
Class Reference . . . . .	2319
6.451.1 Detailed Description . . . . .	2320
6.451.2 Constructor & Destructor Documentation . . . . .	2320
6.451.2.1 JournalTransactionMarshaller . . . . .	2320
6.451.2.2 ~JournalTransactionMarshaller . . . . .	2320
6.451.3 Member Function Documentation . . . . .	2320
6.451.3.1 createObject . . . . .	2320
6.451.3.2 getDataStructureType . . . . .	2321
6.451.3.3 looseMarshal . . . . .	2321
6.451.3.4 looseUnmarshal . . . . .	2321
6.451.3.5 tightMarshal1 . . . . .	2322
6.451.3.6 tightMarshal2 . . . . .	2322
6.451.3.7 tightUnmarshal . . . . .	2323
6.452activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller	
Class Reference . . . . .	2323
6.452.1 Detailed Description . . . . .	2324
6.452.2 Constructor & Destructor Documentation . . . . .	2324
6.452.2.1 JournalTransactionMarshaller . . . . .	2324
6.452.2.2 ~JournalTransactionMarshaller . . . . .	2324
6.452.3 Member Function Documentation . . . . .	2324
6.452.3.1 createObject . . . . .	2324
6.452.3.2 getDataStructureType . . . . .	2325
6.452.3.3 looseMarshal . . . . .	2325
6.452.3.4 looseUnmarshal . . . . .	2325
6.452.3.5 tightMarshal1 . . . . .	2326
6.452.3.6 tightMarshal2 . . . . .	2326
6.452.3.7 tightUnmarshal . . . . .	2327
6.453activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller	
Class Reference . . . . .	2327
6.453.1 Detailed Description . . . . .	2328
6.453.2 Constructor & Destructor Documentation . . . . .	2328
6.453.2.1 JournalTransactionMarshaller . . . . .	2328
6.453.2.2 ~JournalTransactionMarshaller . . . . .	2328
6.453.3 Member Function Documentation . . . . .	2328
6.453.3.1 createObject . . . . .	2328
6.453.3.2 getDataStructureType . . . . .	2329
6.453.3.3 looseMarshal . . . . .	2329

6.453.3.4	looseUnmarshal . . . . .	2329
6.453.3.5	tightMarshal1 . . . . .	2330
6.453.3.6	tightMarshal2 . . . . .	2330
6.453.3.7	tightUnmarshal . . . . .	2331
6.454	activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller	
	Class Reference . . . . .	2331
6.454.1	Detailed Description . . . . .	2332
6.454.2	Constructor & Destructor Documentation . . . . .	2332
6.454.2.1	JournalTransactionMarshaller . . . . .	2332
6.454.2.2	~JournalTransactionMarshaller . . . . .	2332
6.454.3	Member Function Documentation . . . . .	2332
6.454.3.1	createObject . . . . .	2332
6.454.3.2	getDataStructureType . . . . .	2333
6.454.3.3	looseMarshal . . . . .	2333
6.454.3.4	looseUnmarshal . . . . .	2333
6.454.3.5	tightMarshal1 . . . . .	2334
6.454.3.6	tightMarshal2 . . . . .	2334
6.454.3.7	tightUnmarshal . . . . .	2335
6.455	activemq::commands::KeepAliveInfo Class Reference . . . . .	2335
6.455.1	Constructor & Destructor Documentation . . . . .	2336
6.455.1.1	KeepAliveInfo . . . . .	2336
6.455.1.2	~KeepAliveInfo . . . . .	2336
6.455.2	Member Function Documentation . . . . .	2336
6.455.2.1	cloneDataStructure . . . . .	2336
6.455.2.2	copyDataStructure . . . . .	2336
6.455.2.3	equals . . . . .	2337
6.455.2.4	getDataStructureType . . . . .	2337
6.455.2.5	isKeepAliveInfo . . . . .	2337
6.455.2.6	toString . . . . .	2337
6.455.2.7	visit . . . . .	2338
6.455.3	Field Documentation . . . . .	2338
6.455.3.1	ID_KEEPLIVEINFO . . . . .	2338
6.456	activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller	
	Class Reference . . . . .	2338
6.456.1	Detailed Description . . . . .	2339
6.456.2	Constructor & Destructor Documentation . . . . .	2339
6.456.2.1	KeepAliveInfoMarshaller . . . . .	2339
6.456.2.2	~KeepAliveInfoMarshaller . . . . .	2339
6.456.3	Member Function Documentation . . . . .	2339
6.456.3.1	createObject . . . . .	2339
6.456.3.2	getDataStructureType . . . . .	2340
6.456.3.3	looseMarshal . . . . .	2340
6.456.3.4	looseUnmarshal . . . . .	2340
6.456.3.5	tightMarshal1 . . . . .	2341
6.456.3.6	tightMarshal2 . . . . .	2341
6.456.3.7	tightUnmarshal . . . . .	2342
6.457	activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller	
	Class Reference . . . . .	2342
6.457.1	Detailed Description . . . . .	2343
6.457.2	Constructor & Destructor Documentation . . . . .	2344

6.457.2.1	KeepAliveInfoMarshaller	2344
6.457.2.2	~KeepAliveInfoMarshaller	2344
6.457.3	Member Function Documentation	2344
6.457.3.1	createObject	2344
6.457.3.2	getDataStructureType	2344
6.457.3.3	looseMarshal	2344
6.457.3.4	looseUnmarshal	2345
6.457.3.5	tightMarshal1	2345
6.457.3.6	tightMarshal2	2346
6.457.3.7	tightUnmarshal	2346
6.458	activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller	
	Class Reference	2347
6.458.1	Detailed Description	2348
6.458.2	Constructor & Destructor Documentation	2348
6.458.2.1	KeepAliveInfoMarshaller	2348
6.458.2.2	~KeepAliveInfoMarshaller	2348
6.458.3	Member Function Documentation	2348
6.458.3.1	createObject	2348
6.458.3.2	getDataStructureType	2348
6.458.3.3	looseMarshal	2348
6.458.3.4	looseUnmarshal	2349
6.458.3.5	tightMarshal1	2349
6.458.3.6	tightMarshal2	2350
6.458.3.7	tightUnmarshal	2350
6.459	activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller	
	Class Reference	2351
6.459.1	Detailed Description	2352
6.459.2	Constructor & Destructor Documentation	2352
6.459.2.1	KeepAliveInfoMarshaller	2352
6.459.2.2	~KeepAliveInfoMarshaller	2352
6.459.3	Member Function Documentation	2352
6.459.3.1	createObject	2352
6.459.3.2	getDataStructureType	2352
6.459.3.3	looseMarshal	2353
6.459.3.4	looseUnmarshal	2353
6.459.3.5	tightMarshal1	2354
6.459.3.6	tightMarshal2	2354
6.459.3.7	tightUnmarshal	2355
6.460	activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller	
	Class Reference	2355
6.460.1	Detailed Description	2356
6.460.2	Constructor & Destructor Documentation	2356
6.460.2.1	KeepAliveInfoMarshaller	2356
6.460.2.2	~KeepAliveInfoMarshaller	2356
6.460.3	Member Function Documentation	2356
6.460.3.1	createObject	2356
6.460.3.2	getDataStructureType	2357
6.460.3.3	looseMarshal	2357
6.460.3.4	looseUnmarshal	2357
6.460.3.5	tightMarshal1	2358

6.460.3.6	tightMarshal2	2358
6.460.3.7	tightUnmarshal	2359
6.461	activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	
	Class Reference	2359
6.461.1	Detailed Description	2360
6.461.2	Constructor & Destructor Documentation	2361
6.461.2.1	KeepAliveInfoMarshaller	2361
6.461.2.2	~KeepAliveInfoMarshaller	2361
6.461.3	Member Function Documentation	2361
6.461.3.1	createObject	2361
6.461.3.2	getDataStructureType	2361
6.461.3.3	looseMarshal	2361
6.461.3.4	looseUnmarshal	2362
6.461.3.5	tightMarshal1	2362
6.461.3.6	tightMarshal2	2363
6.461.3.7	tightUnmarshal	2363
6.462	decaf::security::Key Class Reference	2364
6.462.1	Detailed Description	2364
6.462.2	Constructor & Destructor Documentation	2365
6.462.2.1	~Key	2365
6.462.3	Member Function Documentation	2365
6.462.3.1	getAlgorithm	2365
6.462.3.2	getEncoded	2365
6.462.3.3	getFormat	2365
6.463	decaf::security::KeyException Class Reference	2366
6.463.1	Constructor & Destructor Documentation	2367
6.463.1.1	KeyException	2367
6.463.1.2	KeyException	2367
6.463.1.3	KeyException	2367
6.463.1.4	KeyException	2367
6.463.1.5	KeyException	2367
6.463.1.6	KeyException	2368
6.463.1.7	~KeyException	2368
6.463.2	Member Function Documentation	2368
6.463.2.1	clone	2368
6.464	decaf::security::KeyManagementException Class Reference	2368
6.464.1	Constructor & Destructor Documentation	2369
6.464.1.1	KeyManagementException	2369
6.464.1.2	KeyManagementException	2369
6.464.1.3	KeyManagementException	2370
6.464.1.4	KeyManagementException	2370
6.464.1.5	KeyManagementException	2370
6.464.1.6	KeyManagementException	2370
6.464.1.7	~KeyManagementException	2371
6.464.2	Member Function Documentation	2371
6.464.2.1	clone	2371
6.465	activemq::commands::LastPartialCommand Class Reference	2371
6.465.1	Constructor & Destructor Documentation	2372
6.465.1.1	LastPartialCommand	2372
6.465.1.2	~LastPartialCommand	2372

6.465.2 Member Function Documentation . . . . .	2372
6.465.2.1 cloneDataStructure . . . . .	2372
6.465.2.2 copyDataStructure . . . . .	2372
6.465.2.3 equals . . . . .	2373
6.465.2.4 getDataStructureType . . . . .	2373
6.465.2.5 toString . . . . .	2373
6.465.3 Field Documentation . . . . .	2373
6.465.3.1 ID_LASTPARTIALCOMMAND . . . . .	2373
6.466activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller	
Class Reference . . . . .	2374
6.466.1 Detailed Description . . . . .	2375
6.466.2 Constructor & Destructor Documentation . . . . .	2375
6.466.2.1 LastPartialCommandMarshaller . . . . .	2375
6.466.2.2 ~LastPartialCommandMarshaller . . . . .	2375
6.466.3 Member Function Documentation . . . . .	2375
6.466.3.1 createObject . . . . .	2375
6.466.3.2 getDataStructureType . . . . .	2375
6.466.3.3 looseMarshal . . . . .	2375
6.466.3.4 looseUnmarshal . . . . .	2376
6.466.3.5 tightMarshal1 . . . . .	2376
6.466.3.6 tightMarshal2 . . . . .	2377
6.466.3.7 tightUnmarshal . . . . .	2377
6.467activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller	
Class Reference . . . . .	2378
6.467.1 Detailed Description . . . . .	2379
6.467.2 Constructor & Destructor Documentation . . . . .	2379
6.467.2.1 LastPartialCommandMarshaller . . . . .	2379
6.467.2.2 ~LastPartialCommandMarshaller . . . . .	2379
6.467.3 Member Function Documentation . . . . .	2379
6.467.3.1 createObject . . . . .	2379
6.467.3.2 getDataStructureType . . . . .	2379
6.467.3.3 looseMarshal . . . . .	2380
6.467.3.4 looseUnmarshal . . . . .	2380
6.467.3.5 tightMarshal1 . . . . .	2381
6.467.3.6 tightMarshal2 . . . . .	2381
6.467.3.7 tightUnmarshal . . . . .	2382
6.468activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller	
Class Reference . . . . .	2382
6.468.1 Detailed Description . . . . .	2383
6.468.2 Constructor & Destructor Documentation . . . . .	2383
6.468.2.1 LastPartialCommandMarshaller . . . . .	2383
6.468.2.2 ~LastPartialCommandMarshaller . . . . .	2383
6.468.3 Member Function Documentation . . . . .	2383
6.468.3.1 createObject . . . . .	2383
6.468.3.2 getDataStructureType . . . . .	2384
6.468.3.3 looseMarshal . . . . .	2384
6.468.3.4 looseUnmarshal . . . . .	2384
6.468.3.5 tightMarshal1 . . . . .	2385
6.468.3.6 tightMarshal2 . . . . .	2385
6.468.3.7 tightUnmarshal . . . . .	2386



6.469	activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller	
	Class Reference . . . . .	2386
6.469.1	Detailed Description . . . . .	2387
6.469.2	Constructor & Destructor Documentation . . . . .	2388
	6.469.2.1 LastPartialCommandMarshaller . . . . .	2388
	6.469.2.2 ~LastPartialCommandMarshaller . . . . .	2388
6.469.3	Member Function Documentation . . . . .	2388
	6.469.3.1 createObject . . . . .	2388
	6.469.3.2 getDataStructureType . . . . .	2388
	6.469.3.3 looseMarshal . . . . .	2388
	6.469.3.4 looseUnmarshal . . . . .	2389
	6.469.3.5 tightMarshal1 . . . . .	2389
	6.469.3.6 tightMarshal2 . . . . .	2390
	6.469.3.7 tightUnmarshal . . . . .	2390
6.470	activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller	
	Class Reference . . . . .	2391
6.470.1	Detailed Description . . . . .	2392
6.470.2	Constructor & Destructor Documentation . . . . .	2392
	6.470.2.1 LastPartialCommandMarshaller . . . . .	2392
	6.470.2.2 ~LastPartialCommandMarshaller . . . . .	2392
6.470.3	Member Function Documentation . . . . .	2392
	6.470.3.1 createObject . . . . .	2392
	6.470.3.2 getDataStructureType . . . . .	2392
	6.470.3.3 looseMarshal . . . . .	2393
	6.470.3.4 looseUnmarshal . . . . .	2393
	6.470.3.5 tightMarshal1 . . . . .	2393
	6.470.3.6 tightMarshal2 . . . . .	2394
	6.470.3.7 tightUnmarshal . . . . .	2394
6.471	activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller	
	Class Reference . . . . .	2395
6.471.1	Detailed Description . . . . .	2396
6.471.2	Constructor & Destructor Documentation . . . . .	2396
	6.471.2.1 LastPartialCommandMarshaller . . . . .	2396
	6.471.2.2 ~LastPartialCommandMarshaller . . . . .	2396
6.471.3	Member Function Documentation . . . . .	2396
	6.471.3.1 createObject . . . . .	2396
	6.471.3.2 getDataStructureType . . . . .	2397
	6.471.3.3 looseMarshal . . . . .	2397
	6.471.3.4 looseUnmarshal . . . . .	2397
	6.471.3.5 tightMarshal1 . . . . .	2398
	6.471.3.6 tightMarshal2 . . . . .	2398
	6.471.3.7 tightUnmarshal . . . . .	2399
6.472	decaf::util::comparators::Less< E > Class Template Reference . . . .	2399
	6.472.1 Detailed Description . . . . .	2400
6.472.2	Constructor & Destructor Documentation . . . . .	2400
	6.472.2.1 Less . . . . .	2400
	6.472.2.2 ~Less . . . . .	2400
6.472.3	Member Function Documentation . . . . .	2400
	6.472.3.1 compare . . . . .	2400
	6.472.3.2 operator() . . . . .	2401

6.473	std::less< decaf::lang::ArrayPointer< T > > Struct Template Reference	2401
6.473.1	Detailed Description	2402
6.473.2	Member Function Documentation	2402
6.473.2.1	operator()	2402
6.474	std::less< decaf::lang::Pointer< T > > Struct Template Reference	2402
6.474.1	Detailed Description	2403
6.474.2	Member Function Documentation	2403
6.474.2.1	operator()	2403
6.475	decaf::util::logging::Level Class Reference	2403
6.475.1	Detailed Description	2405
6.475.2	Constructor & Destructor Documentation	2405
6.475.2.1	Level	2405
6.475.2.2	~Level	2406
6.475.3	Member Function Documentation	2406
6.475.3.1	compareTo	2406
6.475.3.2	equals	2406
6.475.3.3	getName	2406
6.475.3.4	intValue	2406
6.475.3.5	operator<	2406
6.475.3.6	operator==	2406
6.475.3.7	parse	2406
6.475.3.8	toString	2407
6.475.4	Field Documentation	2407
6.475.4.1	ALL	2407
6.475.4.2	CONFIG	2407
6.475.4.3	DEBUG	2407
6.475.4.4	FINE	2407
6.475.4.5	FINER	2408
6.475.4.6	FINEST	2408
6.475.4.7	INFO	2408
6.475.4.8	INHERIT	2408
6.475.4.9	OFF	2408
6.475.4.10	SEVERE	2408
6.475.4.11	WARNING	2408
6.476	decaf::util::List< E > Class Template Reference	2409
6.476.1	Detailed Description	2410
6.476.2	Constructor & Destructor Documentation	2410
6.476.2.1	List	2410
6.476.2.2	~List	2410
6.476.3	Member Function Documentation	2410
6.476.3.1	add	2410
6.476.3.2	addAll	2411
6.476.3.3	get	2412
6.476.3.4	indexOf	2412
6.476.3.5	lastIndexOf	2413
6.476.3.6	listIterator	2414
6.476.3.7	listIterator	2414
6.476.3.8	listIterator	2414
6.476.3.9	listIterator	2415
6.476.3.10	remove	2415

6.476.3.1 lset . . . . .	2416
6.477decaf::util::ListIterator< E > Class Template Reference . . . . .	2417
6.477.1 Detailed Description . . . . .	2417
6.477.2 Constructor & Destructor Documentation . . . . .	2418
6.477.2.1 ~ListIterator . . . . .	2418
6.477.3 Member Function Documentation . . . . .	2418
6.477.3.1 add . . . . .	2418
6.477.3.2 hasPrevious . . . . .	2418
6.477.3.3 nextIndex . . . . .	2419
6.477.3.4 previous . . . . .	2419
6.477.3.5 previousIndex . . . . .	2419
6.477.3.6 set . . . . .	2420
6.478activemq::commands::LocalTransactionId Class Reference . . . . .	2420
6.478.1 Member Typedef Documentation . . . . .	2421
6.478.1.1 COMPARATOR . . . . .	2421
6.478.2 Constructor & Destructor Documentation . . . . .	2422
6.478.2.1 LocalTransactionId . . . . .	2422
6.478.2.2 LocalTransactionId . . . . .	2422
6.478.2.3 ~LocalTransactionId . . . . .	2422
6.478.3 Member Function Documentation . . . . .	2422
6.478.3.1 cloneDataStructure . . . . .	2422
6.478.3.2 compareTo . . . . .	2422
6.478.3.3 copyDataStructure . . . . .	2422
6.478.3.4 equals . . . . .	2422
6.478.3.5 equals . . . . .	2423
6.478.3.6 getConnectionId . . . . .	2423
6.478.3.7 getConnectionId . . . . .	2423
6.478.3.8 getDataStructureType . . . . .	2423
6.478.3.9 getValue . . . . .	2424
6.478.3.10operator< . . . . .	2424
6.478.3.11operator= . . . . .	2424
6.478.3.12operator== . . . . .	2424
6.478.3.13setConnectionId . . . . .	2424
6.478.3.14setValue . . . . .	2424
6.478.3.15toString . . . . .	2424
6.478.4 Field Documentation . . . . .	2424
6.478.4.1 connectionId . . . . .	2424
6.478.4.2 ID_LOCALTRANSACTIONID . . . . .	2424
6.478.4.3 value . . . . .	2424
6.479activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller	
Class Reference . . . . .	2425
6.479.1 Detailed Description . . . . .	2426
6.479.2 Constructor & Destructor Documentation . . . . .	2426
6.479.2.1 LocalTransactionIdMarshaller . . . . .	2426
6.479.2.2 ~LocalTransactionIdMarshaller . . . . .	2426
6.479.3 Member Function Documentation . . . . .	2426
6.479.3.1 createObject . . . . .	2426
6.479.3.2 getDataStructureType . . . . .	2426
6.479.3.3 looseMarshal . . . . .	2426
6.479.3.4 looseUnmarshal . . . . .	2427

6.479.3.5	tightMarshal1	2427
6.479.3.6	tightMarshal2	2428
6.479.3.7	tightUnmarshal	2428
6.480	activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller	
	Class Reference	2429
6.480.1	Detailed Description	2430
6.480.2	Constructor & Destructor Documentation	2430
6.480.2.1	LocalTransactionIdMarshaller	2430
6.480.2.2	~LocalTransactionIdMarshaller	2430
6.480.3	Member Function Documentation	2430
6.480.3.1	createObject	2430
6.480.3.2	getDataStructureType	2430
6.480.3.3	looseMarshal	2431
6.480.3.4	looseUnmarshal	2431
6.480.3.5	tightMarshal1	2432
6.480.3.6	tightMarshal2	2432
6.480.3.7	tightUnmarshal	2433
6.481	activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller	
	Class Reference	2433
6.481.1	Detailed Description	2434
6.481.2	Constructor & Destructor Documentation	2434
6.481.2.1	LocalTransactionIdMarshaller	2434
6.481.2.2	~LocalTransactionIdMarshaller	2434
6.481.3	Member Function Documentation	2434
6.481.3.1	createObject	2434
6.481.3.2	getDataStructureType	2435
6.481.3.3	looseMarshal	2435
6.481.3.4	looseUnmarshal	2435
6.481.3.5	tightMarshal1	2436
6.481.3.6	tightMarshal2	2436
6.481.3.7	tightUnmarshal	2437
6.482	activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller	
	Class Reference	2437
6.482.1	Detailed Description	2438
6.482.2	Constructor & Destructor Documentation	2439
6.482.2.1	LocalTransactionIdMarshaller	2439
6.482.2.2	~LocalTransactionIdMarshaller	2439
6.482.3	Member Function Documentation	2439
6.482.3.1	createObject	2439
6.482.3.2	getDataStructureType	2439
6.482.3.3	looseMarshal	2439
6.482.3.4	looseUnmarshal	2440
6.482.3.5	tightMarshal1	2440
6.482.3.6	tightMarshal2	2441
6.482.3.7	tightUnmarshal	2441
6.483	activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller	
	Class Reference	2442
6.483.1	Detailed Description	2443
6.483.2	Constructor & Destructor Documentation	2443
6.483.2.1	LocalTransactionIdMarshaller	2443

6.483.2.2 ~LocalTransactionIdMarshaller . . . . .	2443
6.483.3 Member Function Documentation . . . . .	2443
6.483.3.1 createObject . . . . .	2443
6.483.3.2 getDataStructureType . . . . .	2443
6.483.3.3 looseMarshal . . . . .	2443
6.483.3.4 looseUnmarshal . . . . .	2444
6.483.3.5 tightMarshal1 . . . . .	2444
6.483.3.6 tightMarshal2 . . . . .	2445
6.483.3.7 tightUnmarshal . . . . .	2445
6.484activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller	
Class Reference . . . . .	2446
6.484.1 Detailed Description . . . . .	2447
6.484.2 Constructor & Destructor Documentation . . . . .	2447
6.484.2.1 LocalTransactionIdMarshaller . . . . .	2447
6.484.2.2 ~LocalTransactionIdMarshaller . . . . .	2447
6.484.3 Member Function Documentation . . . . .	2447
6.484.3.1 createObject . . . . .	2447
6.484.3.2 getDataStructureType . . . . .	2447
6.484.3.3 looseMarshal . . . . .	2448
6.484.3.4 looseUnmarshal . . . . .	2448
6.484.3.5 tightMarshal1 . . . . .	2449
6.484.3.6 tightMarshal2 . . . . .	2449
6.484.3.7 tightUnmarshal . . . . .	2450
6.485decaf::util::concurrent::Lock Class Reference . . . . .	2450
6.485.1 Detailed Description . . . . .	2451
6.485.2 Constructor & Destructor Documentation . . . . .	2451
6.485.2.1 Lock . . . . .	2451
6.485.2.2 ~Lock . . . . .	2451
6.485.3 Member Function Documentation . . . . .	2451
6.485.3.1 isLocked . . . . .	2451
6.485.3.2 lock . . . . .	2451
6.485.3.3 unlock . . . . .	2451
6.486decaf::util::concurrent::locks::Lock Class Reference . . . . .	2452
6.486.1 Detailed Description . . . . .	2452
6.486.2 Constructor & Destructor Documentation . . . . .	2454
6.486.2.1 ~Lock . . . . .	2454
6.486.3 Member Function Documentation . . . . .	2454
6.486.3.1 lock . . . . .	2454
6.486.3.2 lockInterruptibly . . . . .	2454
6.486.3.3 newCondition . . . . .	2455
6.486.3.4 tryLock . . . . .	2456
6.486.3.5 tryLock . . . . .	2457
6.486.3.6 unlock . . . . .	2457
6.487decaf::util::concurrent::locks::LockSupport Class Reference . . . . .	2458
6.487.1 Detailed Description . . . . .	2458
6.487.2 Constructor & Destructor Documentation . . . . .	2459
6.487.2.1 ~LockSupport . . . . .	2459
6.487.3 Member Function Documentation . . . . .	2459
6.487.3.1 park . . . . .	2459
6.487.3.2 parkNanos . . . . .	2460

6.487.3.3 parkUntil . . . . .	2460
6.487.3.4 unpark . . . . .	2461
6.488decaf::util::logging::Logger Class Reference . . . . .	2461
6.488.1 Detailed Description . . . . .	2464
6.488.2 Constructor & Destructor Documentation . . . . .	2465
6.488.2.1 Logger . . . . .	2465
6.488.2.2 ~Logger . . . . .	2465
6.488.3 Member Function Documentation . . . . .	2465
6.488.3.1 addHandler . . . . .	2465
6.488.3.2 config . . . . .	2466
6.488.3.3 debug . . . . .	2466
6.488.3.4 entering . . . . .	2466
6.488.3.5 exiting . . . . .	2467
6.488.3.6 fine . . . . .	2467
6.488.3.7 finer . . . . .	2467
6.488.3.8 finest . . . . .	2468
6.488.3.9 getAnonymousLogger . . . . .	2468
6.488.3.10getFilter . . . . .	2468
6.488.3.11getHandlers . . . . .	2468
6.488.3.12getLevel . . . . .	2469
6.488.3.13getLogger . . . . .	2469
6.488.3.14getName . . . . .	2469
6.488.3.15getParent . . . . .	2469
6.488.3.16getUseParentHandlers . . . . .	2470
6.488.3.17info . . . . .	2470
6.488.3.18isLoggable . . . . .	2470
6.488.3.19log . . . . .	2470
6.488.3.20log . . . . .	2471
6.488.3.21log . . . . .	2471
6.488.3.22log . . . . .	2471
6.488.3.23removeHandler . . . . .	2472
6.488.3.24setFilter . . . . .	2472
6.488.3.25setLevel . . . . .	2472
6.488.3.26setParent . . . . .	2472
6.488.3.27setUseParentHandlers . . . . .	2473
6.488.3.28severe . . . . .	2473
6.488.3.29throwing . . . . .	2473
6.488.3.30warning . . . . .	2474
6.489decaf::util::logging::LoggerHierarchy Class Reference . . . . .	2474
6.489.1 Constructor & Destructor Documentation . . . . .	2474
6.489.1.1 LoggerHierarchy . . . . .	2474
6.489.1.2 ~LoggerHierarchy . . . . .	2474
6.490activemq::io::LoggingInputStream Class Reference . . . . .	2474
6.490.1 Constructor & Destructor Documentation . . . . .	2475
6.490.1.1 LoggingInputStream . . . . .	2475
6.490.1.2 ~LoggingInputStream . . . . .	2475
6.490.2 Member Function Documentation . . . . .	2475
6.490.2.1 doReadArrayBounded . . . . .	2475
6.490.2.2 doReadByte . . . . .	2475
6.491activemq::io::LoggingOutputStream Class Reference . . . . .	2476

6.491.1 Detailed Description . . . . .	2476
6.491.2 Constructor & Destructor Documentation . . . . .	2476
6.491.2.1 LoggingOutputStream . . . . .	2476
6.491.2.2 ~LoggingOutputStream . . . . .	2477
6.491.3 Member Function Documentation . . . . .	2477
6.491.3.1 doWriteArrayBounded . . . . .	2477
6.491.3.2 doWriteByte . . . . .	2477
6.492activemq::transport::logging::LoggingTransport Class Reference . . .	2477
6.492.1 Detailed Description . . . . .	2478
6.492.2 Constructor & Destructor Documentation . . . . .	2478
6.492.2.1 LoggingTransport . . . . .	2478
6.492.2.2 ~LoggingTransport . . . . .	2478
6.492.3 Member Function Documentation . . . . .	2478
6.492.3.1 onCommand . . . . .	2478
6.492.3.2 oneway . . . . .	2479
6.492.3.3 request . . . . .	2479
6.492.3.4 request . . . . .	2479
6.493decaf::util::logging::LogManager Class Reference . . . . .	2480
6.493.1 Detailed Description . . . . .	2482
6.493.2 Constructor & Destructor Documentation . . . . .	2483
6.493.2.1 ~LogManager . . . . .	2483
6.493.2.2 LogManager . . . . .	2483
6.493.2.3 LogManager . . . . .	2483
6.493.3 Member Function Documentation . . . . .	2483
6.493.3.1 addLogger . . . . .	2483
6.493.3.2 addPropertyChangeListener . . . . .	2484
6.493.3.3 getLogger . . . . .	2484
6.493.3.4 getLoggerNames . . . . .	2484
6.493.3.5 getLogManager . . . . .	2485
6.493.3.6 getProperties . . . . .	2485
6.493.3.7 getProperty . . . . .	2485
6.493.3.8 operator= . . . . .	2485
6.493.3.9 readConfiguration . . . . .	2485
6.493.3.10readConfiguration . . . . .	2486
6.493.3.11removePropertyChangeListener . . . . .	2486
6.493.3.12reset . . . . .	2486
6.493.3.13setProperties . . . . .	2487
6.493.4 Friends And Related Function Documentation . . . . .	2487
6.493.4.1 decaf::lang::Runtime . . . . .	2487
6.494decaf::util::logging::LogRecord Class Reference . . . . .	2487
6.494.1 Detailed Description . . . . .	2489
6.494.2 Constructor & Destructor Documentation . . . . .	2489
6.494.2.1 LogRecord . . . . .	2489
6.494.2.2 ~LogRecord . . . . .	2489
6.494.3 Member Function Documentation . . . . .	2489
6.494.3.1 getLevel . . . . .	2489
6.494.3.2 getLoggerName . . . . .	2489
6.494.3.3 getMessage . . . . .	2489
6.494.3.4 getSourceFile . . . . .	2490
6.494.3.5 getSourceFunction . . . . .	2490

6.494.3.6	getSourceLine	2490
6.494.3.7	getThrown	2490
6.494.3.8	getTimestamp	2490
6.494.3.9	getTreadId	2491
6.494.3.10	setLevel	2491
6.494.3.11	setLoggerName	2491
6.494.3.12	setMessage	2491
6.494.3.13	setSourceFile	2491
6.494.3.14	setSourceFunction	2492
6.494.3.15	setSourceLine	2492
6.494.3.16	setThrown	2492
6.494.3.17	setTimestamp	2492
6.494.3.18	setTreadId	2492
6.495	decaf::util::logging::LogWriter Class Reference	2493
6.495.1	Constructor & Destructor Documentation	2494
6.495.1.1	LogWriter	2494
6.495.1.2	~LogWriter	2494
6.495.2	Member Function Documentation	2494
6.495.2.1	destroy	2494
6.495.2.2	getInstance	2494
6.495.2.3	log	2494
6.495.2.4	log	2494
6.495.2.5	returnInstance	2494
6.496	decaf::lang::Long Class Reference	2495
6.496.1	Constructor & Destructor Documentation	2498
6.496.1.1	Long	2498
6.496.1.2	Long	2498
6.496.1.3	~Long	2498
6.496.2	Member Function Documentation	2498
6.496.2.1	bitCount	2498
6.496.2.2	byteValue	2499
6.496.2.3	compareTo	2499
6.496.2.4	compareTo	2499
6.496.2.5	decode	2500
6.496.2.6	doubleValue	2500
6.496.2.7	equals	2500
6.496.2.8	equals	2501
6.496.2.9	floatValue	2501
6.496.2.10	highestOneBit	2501
6.496.2.11	intValue	2501
6.496.2.12	longValue	2502
6.496.2.13	lowestOneBit	2502
6.496.2.14	numberOfLeadingZeros	2502
6.496.2.15	numberOfTrailingZeros	2503
6.496.2.16	operator<	2503
6.496.2.17	operator<	2503
6.496.2.18	operator==	2504
6.496.2.19	operator==	2504
6.496.2.20	parseLong	2504
6.496.2.21	parseLong	2505



6.496.2.22	reverse	2505
6.496.2.23	reverseBytes	2506
6.496.2.24	rotateLeft	2506
6.496.2.25	rotateRight	2506
6.496.2.26	shortValue	2507
6.496.2.27	signum	2507
6.496.2.28	toBinaryString	2507
6.496.2.29	toHexString	2508
6.496.2.30	toOctalString	2508
6.496.2.3	toString	2509
6.496.2.32	toString	2509
6.496.2.33	toString	2509
6.496.2.34	valueOf	2509
6.496.2.35	valueOf	2509
6.496.2.36	valueOf	2510
6.496.3	Field Documentation	2510
6.496.3.1	MAX_VALUE	2510
6.496.3.2	MIN_VALUE	2510
6.496.3.3	SIZE	2510
6.497	decaf::internal::nio::LongArrayBuffer Class Reference	2511
6.497.1	Constructor & Destructor Documentation	2515
6.497.1.1	LongArrayBuffer	2515
6.497.1.2	LongArrayBuffer	2515
6.497.1.3	LongArrayBuffer	2516
6.497.1.4	LongArrayBuffer	2516
6.497.1.5	~LongArrayBuffer	2517
6.497.2	Member Function Documentation	2517
6.497.2.1	array	2517
6.497.2.2	arrayOffset	2517
6.497.2.3	asReadOnlyBuffer	2518
6.497.2.4	compact	2518
6.497.2.5	duplicate	2519
6.497.2.6	get	2519
6.497.2.7	get	2519
6.497.2.8	hasArray	2520
6.497.2.9	isReadOnly	2520
6.497.2.10	put	2520
6.497.2.11	put	2521
6.497.2.12	setReadOnly	2521
6.497.2.13	slice	2522
6.498	decaf::nio::LongBuffer Class Reference	2522
6.498.1	Detailed Description	2524
6.498.2	Constructor & Destructor Documentation	2525
6.498.2.1	LongBuffer	2525
6.498.2.2	~LongBuffer	2525
6.498.3	Member Function Documentation	2525
6.498.3.1	allocate	2525
6.498.3.2	array	2526
6.498.3.3	arrayOffset	2526
6.498.3.4	asReadOnlyBuffer	2527

6.498.3.5 compact	2527
6.498.3.6 compareTo	2528
6.498.3.7 duplicate	2528
6.498.3.8 equals	2528
6.498.3.9 get	2528
6.498.3.10get	2528
6.498.3.11get	2529
6.498.3.12get	2530
6.498.3.13hasArray	2530
6.498.3.14operator<	2530
6.498.3.15operator==	2530
6.498.3.16put	2530
6.498.3.17put	2531
6.498.3.18put	2531
6.498.3.19put	2532
6.498.3.20put	2533
6.498.3.21slice	2533
6.498.3.22toString	2534
6.498.3.23wrap	2534
6.498.3.24wrap	2534
6.499activemq::util::LongSequenceGenerator Class Reference	2535
6.499.1 Detailed Description	2535
6.499.2 Constructor & Destructor Documentation	2535
6.499.2.1 LongSequenceGenerator	2535
6.499.2.2 ~LongSequenceGenerator	2535
6.499.3 Member Function Documentation	2535
6.499.3.1 getLastSequenceId	2535
6.499.3.2 getNextSequenceId	2536
6.500decaf::net::MalformedURLException Class Reference	2536
6.500.1 Constructor & Destructor Documentation	2537
6.500.1.1 MalformedURLException	2537
6.500.1.2 MalformedURLException	2537
6.500.1.3 MalformedURLException	2537
6.500.1.4 MalformedURLException	2537
6.500.1.5 MalformedURLException	2537
6.500.1.6 MalformedURLException	2538
6.500.1.7 ~MalformedURLException	2538
6.500.2 Member Function Documentation	2538
6.500.2.1 clone	2538
6.501decaf::util::Map< K, V, COMPARATOR > Class Template Reference	2538
6.501.1 Detailed Description	2540
6.501.2 Constructor & Destructor Documentation	2540
6.501.2.1 Map	2540
6.501.2.2 ~Map	2540
6.501.3 Member Function Documentation	2540
6.501.3.1 clear	2540
6.501.3.2 containsKey	2541
6.501.3.3 containsValue	2542
6.501.3.4 copy	2543
6.501.3.5 equals	2543

6.501.3.6	get . . . . .	2543
6.501.3.7	get . . . . .	2544
6.501.3.8	isEmpty . . . . .	2545
6.501.3.9	keySet . . . . .	2546
6.501.3.10	put . . . . .	2547
6.501.3.11	putAll . . . . .	2548
6.501.3.12	remove . . . . .	2548
6.501.3.13	size . . . . .	2549
6.501.3.14	values . . . . .	2550
6.502	cms::MapMessage Class Reference . . . . .	2551
6.502.1	Detailed Description . . . . .	2553
6.502.2	Constructor & Destructor Documentation . . . . .	2554
6.502.2.1	~MapMessage . . . . .	2554
6.502.3	Member Function Documentation . . . . .	2554
6.502.3.1	getBoolean . . . . .	2554
6.502.3.2	getByte . . . . .	2554
6.502.3.3	getBytes . . . . .	2555
6.502.3.4	getChar . . . . .	2555
6.502.3.5	getDouble . . . . .	2555
6.502.3.6	getFloat . . . . .	2556
6.502.3.7	getInt . . . . .	2556
6.502.3.8	getLong . . . . .	2557
6.502.3.9	getMapNames . . . . .	2557
6.502.3.10	getShort . . . . .	2557
6.502.3.11	getString . . . . .	2558
6.502.3.12	itemExists . . . . .	2558
6.502.3.13	setBoolean . . . . .	2559
6.502.3.14	setByte . . . . .	2559
6.502.3.15	setBytes . . . . .	2559
6.502.3.16	setChar . . . . .	2560
6.502.3.17	setDouble . . . . .	2560
6.502.3.18	setFloat . . . . .	2561
6.502.3.19	setInt . . . . .	2561
6.502.3.20	setLong . . . . .	2562
6.502.3.21	setShort . . . . .	2562
6.502.3.22	setString . . . . .	2562
6.503	decaf::util::logging::MarkBlockLogger Class Reference . . . . .	2563
6.503.1	Detailed Description . . . . .	2563
6.503.2	Constructor & Destructor Documentation . . . . .	2563
6.503.2.1	MarkBlockLogger . . . . .	2563
6.503.2.2	~MarkBlockLogger . . . . .	2564
6.504	activemq::wireformat::MarshalAware Class Reference . . . . .	2564
6.504.1	Constructor & Destructor Documentation . . . . .	2565
6.504.1.1	~MarshalAware . . . . .	2565
6.504.2	Member Function Documentation . . . . .	2565
6.504.2.1	afterMarshal . . . . .	2565
6.504.2.2	afterUnmarshal . . . . .	2565
6.504.2.3	beforeMarshal . . . . .	2565
6.504.2.4	beforeUnmarshal . . . . .	2566
6.504.2.5	getMarshaledForm . . . . .	2566

6.504.2.6 isMarshalAware . . . . .	2566
6.504.2.7 setMarshaledForm . . . . .	2566
6.505activemq::wireformat::openwire::marshal::v6::MarshallerFactory Class	
Reference . . . . .	2567
6.505.1 Detailed Description . . . . .	2567
6.505.2 Constructor & Destructor Documentation . . . . .	2567
6.505.2.1 ~MarshallerFactory . . . . .	2567
6.505.3 Member Function Documentation . . . . .	2567
6.505.3.1 configure . . . . .	2567
6.506activemq::wireformat::openwire::marshal::v3::MarshallerFactory Class	
Reference . . . . .	2567
6.506.1 Detailed Description . . . . .	2568
6.506.2 Constructor & Destructor Documentation . . . . .	2568
6.506.2.1 ~MarshallerFactory . . . . .	2568
6.506.3 Member Function Documentation . . . . .	2568
6.506.3.1 configure . . . . .	2568
6.507activemq::wireformat::openwire::marshal::v4::MarshallerFactory Class	
Reference . . . . .	2568
6.507.1 Detailed Description . . . . .	2568
6.507.2 Constructor & Destructor Documentation . . . . .	2569
6.507.2.1 ~MarshallerFactory . . . . .	2569
6.507.3 Member Function Documentation . . . . .	2569
6.507.3.1 configure . . . . .	2569
6.508activemq::wireformat::openwire::marshal::v5::MarshallerFactory Class	
Reference . . . . .	2569
6.508.1 Detailed Description . . . . .	2569
6.508.2 Constructor & Destructor Documentation . . . . .	2569
6.508.2.1 ~MarshallerFactory . . . . .	2569
6.508.3 Member Function Documentation . . . . .	2569
6.508.3.1 configure . . . . .	2569
6.509activemq::wireformat::openwire::marshal::v1::MarshallerFactory Class	
Reference . . . . .	2570
6.509.1 Detailed Description . . . . .	2570
6.509.2 Constructor & Destructor Documentation . . . . .	2570
6.509.2.1 ~MarshallerFactory . . . . .	2570
6.509.3 Member Function Documentation . . . . .	2570
6.509.3.1 configure . . . . .	2570
6.510activemq::wireformat::openwire::marshal::v2::MarshallerFactory Class	
Reference . . . . .	2570
6.510.1 Detailed Description . . . . .	2571
6.510.2 Constructor & Destructor Documentation . . . . .	2571
6.510.2.1 ~MarshallerFactory . . . . .	2571
6.510.3 Member Function Documentation . . . . .	2571
6.510.3.1 configure . . . . .	2571
6.511activemq::util::MarshallingSupport Class Reference . . . . .	2571
6.511.1 Constructor & Destructor Documentation . . . . .	2572
6.511.1.1 MarshallingSupport . . . . .	2572
6.511.1.2 ~MarshallingSupport . . . . .	2572
6.511.2 Member Function Documentation . . . . .	2572
6.511.2.1 asciiToModifiedUtf8 . . . . .	2572

6.511.2.2	modifiedUtf8ToAscii	2573
6.511.2.3	readString16	2573
6.511.2.4	readString32	2574
6.511.2.5	writeString	2574
6.511.2.6	writeString16	2574
6.511.2.7	writeString32	2575
6.512	decaf::lang::Math Class Reference	2575
6.512.1	Detailed Description	2577
6.512.2	Constructor & Destructor Documentation	2578
6.512.2.1	Math	2578
6.512.2.2	~Math	2578
6.512.3	Member Function Documentation	2578
6.512.3.1	abs	2578
6.512.3.2	abs	2578
6.512.3.3	abs	2578
6.512.3.4	abs	2579
6.512.3.5	ceil	2579
6.512.3.6	floor	2580
6.512.3.7	max	2581
6.512.3.8	max	2581
6.512.3.9	max	2581
6.512.3.10	max	2582
6.512.3.11	lmax	2582
6.512.3.12	min	2582
6.512.3.13	min	2583
6.512.3.14	min	2583
6.512.3.15	min	2583
6.512.3.16	min	2584
6.512.3.17	min	2584
6.512.3.18	pow	2585
6.512.3.19	random	2585
6.512.3.20	round	2586
6.512.3.21	round	2586
6.512.3.22	signum	2587
6.512.3.23	signum	2587
6.512.3.24	sqrt	2588
6.512.3.25	toDegrees	2592
6.512.3.26	toRadians	2592
6.512.4	Field Documentation	2592
6.512.4.1	E	2592
6.512.4.2	PI	2592
6.513	activemq::util::MemoryUsage Class Reference	2592
6.513.1	Constructor & Destructor Documentation	2593
6.513.1.1	MemoryUsage	2593
6.513.1.2	MemoryUsage	2594
6.513.1.3	~MemoryUsage	2594
6.513.2	Member Function Documentation	2594
6.513.2.1	decreaseUsage	2594
6.513.2.2	enqueueUsage	2594
6.513.2.3	getLimit	2594

6.513.2.4	getUsage . . . . .	2594
6.513.2.5	increaseUsage . . . . .	2595
6.513.2.6	isFull . . . . .	2595
6.513.2.7	setLimit . . . . .	2595
6.513.2.8	setUsage . . . . .	2595
6.513.2.9	waitForSpace . . . . .	2595
6.513.2.10	waitForSpace . . . . .	2596
6.514	activemq::commands::Message Class Reference . . . . .	2596
6.514.1	Constructor & Destructor Documentation . . . . .	2601
6.514.1.1	Message . . . . .	2601
6.514.1.2	~Message . . . . .	2601
6.514.2	Member Function Documentation . . . . .	2601
6.514.2.1	afterUnmarshal . . . . .	2601
6.514.2.2	beforeMarshal . . . . .	2601
6.514.2.3	cloneDataStructure . . . . .	2601
6.514.2.4	copyDataStructure . . . . .	2602
6.514.2.5	equals . . . . .	2602
6.514.2.6	getAckHandler . . . . .	2603
6.514.2.7	getArrival . . . . .	2603
6.514.2.8	getBrokerInTime . . . . .	2603
6.514.2.9	getBrokerOutTime . . . . .	2603
6.514.2.10	getBrokerPath . . . . .	2603
6.514.2.11	lgetBrokerPath . . . . .	2603
6.514.2.12	getCluster . . . . .	2603
6.514.2.13	getCluster . . . . .	2603
6.514.2.14	getConnection . . . . .	2603
6.514.2.15	getContent . . . . .	2604
6.514.2.16	getContent . . . . .	2604
6.514.2.17	getCorrelationId . . . . .	2604
6.514.2.18	getCorrelationId . . . . .	2604
6.514.2.19	getDataStructure . . . . .	2604
6.514.2.20	getDataStructure . . . . .	2604
6.514.2.21	lgetDataStructureType . . . . .	2604
6.514.2.22	getDestination . . . . .	2605
6.514.2.23	getDestination . . . . .	2605
6.514.2.24	getExpiration . . . . .	2605
6.514.2.25	getGroupID . . . . .	2605
6.514.2.26	getGroupID . . . . .	2605
6.514.2.27	getGroupSequence . . . . .	2605
6.514.2.28	getMarshaledProperties . . . . .	2605
6.514.2.29	getMarshaledProperties . . . . .	2605
6.514.2.30	getMessageId . . . . .	2605
6.514.2.31	lgetMessageId . . . . .	2605
6.514.2.32	getMessageProperties . . . . .	2605
6.514.2.33	getMessageProperties . . . . .	2606
6.514.2.34	getOriginalDestination . . . . .	2606
6.514.2.35	getOriginalDestination . . . . .	2606
6.514.2.36	getOriginalTransactionId . . . . .	2606
6.514.2.37	getOriginalTransactionId . . . . .	2606
6.514.2.38	getPriority . . . . .	2606

6.514.2.39	getProducerId	2606
6.514.2.40	getProducerId	2606
6.514.2.41	getRedeliveryCounter	2606
6.514.2.42	getReplyTo	2606
6.514.2.43	getReplyTo	2606
6.514.2.44	getSize	2606
6.514.2.45	getTargetConsumerId	2607
6.514.2.46	getTargetConsumerId	2607
6.514.2.47	getTimestamp	2607
6.514.2.48	getTransactionId	2607
6.514.2.49	getTransactionId	2607
6.514.2.50	getType	2607
6.514.2.51	getType	2607
6.514.2.52	getUserID	2607
6.514.2.53	getUserID	2607
6.514.2.54	isCompressed	2607
6.514.2.55	isDroppable	2607
6.514.2.56	isExpired	2607
6.514.2.57	isMarshalAware	2607
6.514.2.58	isMessage	2608
6.514.2.59	isPersistent	2608
6.514.2.60	isReadOnlyBody	2608
6.514.2.61	isReadOnlyProperties	2608
6.514.2.62	isRecievedByDFBridge	2608
6.514.2.63	onSend	2608
6.514.2.64	setAckHandler	2609
6.514.2.65	setArrival	2609
6.514.2.66	setBrokerInTime	2609
6.514.2.67	setBrokerOutTime	2609
6.514.2.68	setBrokerPath	2609
6.514.2.69	setCluster	2609
6.514.2.70	setCompressed	2609
6.514.2.71	setConnection	2609
6.514.2.72	setContent	2610
6.514.2.73	setCorrelationId	2610
6.514.2.74	setDataStructure	2610
6.514.2.75	setDestination	2610
6.514.2.76	setDroppable	2610
6.514.2.77	setExpiration	2610
6.514.2.78	setGroupID	2610
6.514.2.79	setGroupSequence	2610
6.514.2.80	setMarshaledProperties	2610
6.514.2.81	setMessageId	2610
6.514.2.82	setOriginalDestination	2610
6.514.2.83	setOriginalTransactionId	2610
6.514.2.84	setPersistent	2610
6.514.2.85	setPriority	2610
6.514.2.86	setProducerId	2610
6.514.2.87	setReadOnlyBody	2610
6.514.2.88	setReadOnlyProperties	2611

6.514.2.89	setRecievedByDFBridge	2611
6.514.2.90	setRedeliveryCounter	2611
6.514.2.91	setReplyTo	2611
6.514.2.92	setTargetConsumerId	2611
6.514.2.93	setTimestamp	2611
6.514.2.94	setTransactionId	2611
6.514.2.95	setType	2611
6.514.2.96	setUserID	2611
6.514.2.97	toString	2611
6.514.2.98	visit	2612
6.514.3	Field Documentation	2613
6.514.3.1	arrival	2613
6.514.3.2	brokerInTime	2613
6.514.3.3	brokerOutTime	2613
6.514.3.4	brokerPath	2613
6.514.3.5	cluster	2613
6.514.3.6	compressed	2613
6.514.3.7	connection	2613
6.514.3.8	content	2613
6.514.3.9	correlationId	2613
6.514.3.10	dataStructure	2613
6.514.3.11	DEFAULT_MESSAGE_SIZE	2613
6.514.3.12	destination	2613
6.514.3.13	droppable	2613
6.514.3.14	expiration	2613
6.514.3.15	groupId	2613
6.514.3.16	groupSequence	2613
6.514.3.17	ID_MESSAGE	2613
6.514.3.18	marshalledProperties	2613
6.514.3.19	messageId	2613
6.514.3.20	originalDestination	2613
6.514.3.21	originalTransactionId	2613
6.514.3.22	persistent	2613
6.514.3.23	priority	2613
6.514.3.24	producerId	2613
6.514.3.25	recievedByDFBridge	2613
6.514.3.26	redeliveryCounter	2613
6.514.3.27	replyTo	2613
6.514.3.28	targetConsumerId	2613
6.514.3.29	timestamp	2613
6.514.3.30	transactionId	2613
6.514.3.31	type	2613
6.514.3.32	userId	2613
6.515	cms::Message Class Reference	2614
6.515.1	Detailed Description	2617
6.515.2	Constructor & Destructor Documentation	2619
6.515.2.1	~Message	2619
6.515.3	Member Function Documentation	2619
6.515.3.1	acknowledge	2619
6.515.3.2	clearBody	2619



6.515.3.3 clearProperties . . . . .	2620
6.515.3.4 clone . . . . .	2620
6.515.3.5 getBooleanProperty . . . . .	2621
6.515.3.6 getByteProperty . . . . .	2621
6.515.3.7 getCMSCorrelationID . . . . .	2622
6.515.3.8 getCMSDeliveryMode . . . . .	2622
6.515.3.9 getCMSDestination . . . . .	2623
6.515.3.10 getCMSExpiration . . . . .	2623
6.515.3.11 getCMSMessageID . . . . .	2624
6.515.3.12 getCMSPriority . . . . .	2625
6.515.3.13 getCMSRedelivered . . . . .	2625
6.515.3.14 getCMSReplyTo . . . . .	2626
6.515.3.15 getCMSTimestamp . . . . .	2626
6.515.3.16 getCMSType . . . . .	2627
6.515.3.17 getDoubleProperty . . . . .	2628
6.515.3.18 getFloatProperty . . . . .	2628
6.515.3.19 getIntProperty . . . . .	2629
6.515.3.20 getLongProperty . . . . .	2629
6.515.3.21 getPropertyNames . . . . .	2630
6.515.3.22 getShortProperty . . . . .	2631
6.515.3.23 getStringProperty . . . . .	2631
6.515.3.24 propertyExists . . . . .	2632
6.515.3.25 setBooleanProperty . . . . .	2632
6.515.3.26 setByteProperty . . . . .	2633
6.515.3.27 setCMSCorrelationID . . . . .	2633
6.515.3.28 setCMSDeliveryMode . . . . .	2634
6.515.3.29 setCMSDestination . . . . .	2635
6.515.3.30 setCMSExpiration . . . . .	2635
6.515.3.31 setCMSMessageID . . . . .	2636
6.515.3.32 setCMSPriority . . . . .	2636
6.515.3.33 setCMSRedelivered . . . . .	2637
6.515.3.34 setCMSReplyTo . . . . .	2637
6.515.3.35 setCMSTimestamp . . . . .	2638
6.515.3.36 setCMSType . . . . .	2638
6.515.3.37 setDoubleProperty . . . . .	2639
6.515.3.38 setFloatProperty . . . . .	2640
6.515.3.39 setIntProperty . . . . .	2640
6.515.3.40 setLongProperty . . . . .	2641
6.515.3.41 setShortProperty . . . . .	2641
6.515.3.42 setStringProperty . . . . .	2642
6.516 activemq::commands::MessageAck Class Reference . . . . .	2643
6.516.1 Constructor & Destructor Documentation . . . . .	2644
6.516.1.1 MessageAck . . . . .	2644
6.516.1.2 ~MessageAck . . . . .	2644
6.516.2 Member Function Documentation . . . . .	2644
6.516.2.1 cloneDataStructure . . . . .	2644
6.516.2.2 copyDataStructure . . . . .	2645
6.516.2.3 equals . . . . .	2645
6.516.2.4 getAckType . . . . .	2645
6.516.2.5 getConsumerId . . . . .	2645

6.516.2.6	getConsumerId	2645
6.516.2.7	getDataStructureType	2645
6.516.2.8	getDestination	2646
6.516.2.9	getDestination	2646
6.516.2.10	getFirstMessageId	2646
6.516.2.11	getFirstMessageId	2646
6.516.2.12	getLastMessageId	2646
6.516.2.13	getLastMessageId	2646
6.516.2.14	getMessageCount	2646
6.516.2.15	getTransactionId	2646
6.516.2.16	getTransactionId	2646
6.516.2.17	isMessageAck	2646
6.516.2.18	setAckType	2647
6.516.2.19	setConsumerId	2647
6.516.2.20	setDestination	2647
6.516.2.21	setFirstMessageId	2647
6.516.2.22	setLastMessageId	2647
6.516.2.23	setMessageCount	2647
6.516.2.24	setTransactionId	2647
6.516.2.25	toString	2647
6.516.2.26	visit	2647
6.516.3	Field Documentation	2648
6.516.3.1	ackType	2648
6.516.3.2	consumerId	2648
6.516.3.3	destination	2648
6.516.3.4	firstMessageId	2648
6.516.3.5	ID_MESSAGEACK	2648
6.516.3.6	lastMessageId	2648
6.516.3.7	messageCount	2648
6.516.3.8	transactionId	2648
6.517	activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller	
	Class Reference	2648
6.517.1	Detailed Description	2649
6.517.2	Constructor & Destructor Documentation	2650
6.517.2.1	MessageAckMarshaller	2650
6.517.2.2	~MessageAckMarshaller	2650
6.517.3	Member Function Documentation	2650
6.517.3.1	createObject	2650
6.517.3.2	getDataStructureType	2650
6.517.3.3	looseMarshal	2650
6.517.3.4	looseUnmarshal	2651
6.517.3.5	tightMarshal1	2651
6.517.3.6	tightMarshal2	2652
6.517.3.7	tightUnmarshal	2652
6.518	activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller	
	Class Reference	2653
6.518.1	Detailed Description	2654
6.518.2	Constructor & Destructor Documentation	2654
6.518.2.1	MessageAckMarshaller	2654
6.518.2.2	~MessageAckMarshaller	2654

6.518.3 Member Function Documentation . . . . .	2654
6.518.3.1 createObject . . . . .	2654
6.518.3.2 getDataStructureType . . . . .	2654
6.518.3.3 looseMarshal . . . . .	2654
6.518.3.4 looseUnmarshal . . . . .	2655
6.518.3.5 tightMarshal1 . . . . .	2655
6.518.3.6 tightMarshal2 . . . . .	2656
6.518.3.7 tightUnmarshal . . . . .	2656
6.519activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller	
Class Reference . . . . .	2657
6.519.1 Detailed Description . . . . .	2658
6.519.2 Constructor & Destructor Documentation . . . . .	2658
6.519.2.1 MessageAckMarshaller . . . . .	2658
6.519.2.2 ~MessageAckMarshaller . . . . .	2658
6.519.3 Member Function Documentation . . . . .	2658
6.519.3.1 createObject . . . . .	2658
6.519.3.2 getDataStructureType . . . . .	2658
6.519.3.3 looseMarshal . . . . .	2659
6.519.3.4 looseUnmarshal . . . . .	2659
6.519.3.5 tightMarshal1 . . . . .	2660
6.519.3.6 tightMarshal2 . . . . .	2660
6.519.3.7 tightUnmarshal . . . . .	2661
6.520activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller	
Class Reference . . . . .	2661
6.520.1 Detailed Description . . . . .	2662
6.520.2 Constructor & Destructor Documentation . . . . .	2662
6.520.2.1 MessageAckMarshaller . . . . .	2662
6.520.2.2 ~MessageAckMarshaller . . . . .	2662
6.520.3 Member Function Documentation . . . . .	2662
6.520.3.1 createObject . . . . .	2662
6.520.3.2 getDataStructureType . . . . .	2663
6.520.3.3 looseMarshal . . . . .	2663
6.520.3.4 looseUnmarshal . . . . .	2663
6.520.3.5 tightMarshal1 . . . . .	2664
6.520.3.6 tightMarshal2 . . . . .	2664
6.520.3.7 tightUnmarshal . . . . .	2665
6.521activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller	
Class Reference . . . . .	2665
6.521.1 Detailed Description . . . . .	2666
6.521.2 Constructor & Destructor Documentation . . . . .	2667
6.521.2.1 MessageAckMarshaller . . . . .	2667
6.521.2.2 ~MessageAckMarshaller . . . . .	2667
6.521.3 Member Function Documentation . . . . .	2667
6.521.3.1 createObject . . . . .	2667
6.521.3.2 getDataStructureType . . . . .	2667
6.521.3.3 looseMarshal . . . . .	2667
6.521.3.4 looseUnmarshal . . . . .	2668
6.521.3.5 tightMarshal1 . . . . .	2668
6.521.3.6 tightMarshal2 . . . . .	2669
6.521.3.7 tightUnmarshal . . . . .	2669

6.522activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller	
Class Reference . . . . .	2670
6.522.1 Detailed Description . . . . .	2671
6.522.2 Constructor & Destructor Documentation . . . . .	2671
6.522.2.1 MessageAckMarshaller . . . . .	2671
6.522.2.2 ~MessageAckMarshaller . . . . .	2671
6.522.3 Member Function Documentation . . . . .	2671
6.522.3.1 createObject . . . . .	2671
6.522.3.2 getDataStructureType . . . . .	2671
6.522.3.3 looseMarshal . . . . .	2671
6.522.3.4 looseUnmarshal . . . . .	2672
6.522.3.5 tightMarshal1 . . . . .	2672
6.522.3.6 tightMarshal2 . . . . .	2673
6.522.3.7 tightUnmarshal . . . . .	2673
6.523cms::MessageConsumer Class Reference . . . . .	2674
6.523.1 Detailed Description . . . . .	2675
6.523.2 Constructor & Destructor Documentation . . . . .	2675
6.523.2.1 ~MessageConsumer . . . . .	2675
6.523.3 Member Function Documentation . . . . .	2675
6.523.3.1 getMessageListener . . . . .	2675
6.523.3.2 getMessageSelector . . . . .	2676
6.523.3.3 receive . . . . .	2676
6.523.3.4 receive . . . . .	2676
6.523.3.5 receiveNoWait . . . . .	2677
6.523.3.6 setMessageListener . . . . .	2677
6.524activemq::cmsutil::MessageCreator Class Reference . . . . .	2677
6.524.1 Detailed Description . . . . .	2678
6.524.2 Constructor & Destructor Documentation . . . . .	2678
6.524.2.1 ~MessageCreator . . . . .	2678
6.524.3 Member Function Documentation . . . . .	2678
6.524.3.1 createMessage . . . . .	2678
6.525activemq::commands::MessageDispatch Class Reference . . . . .	2678
6.525.1 Constructor & Destructor Documentation . . . . .	2680
6.525.1.1 MessageDispatch . . . . .	2680
6.525.1.2 ~MessageDispatch . . . . .	2680
6.525.2 Member Function Documentation . . . . .	2680
6.525.2.1 cloneDataStructure . . . . .	2680
6.525.2.2 copyDataStructure . . . . .	2680
6.525.2.3 equals . . . . .	2680
6.525.2.4 getConsumerId . . . . .	2681
6.525.2.5 getConsumerId . . . . .	2681
6.525.2.6 getDataStructureType . . . . .	2681
6.525.2.7 getDestination . . . . .	2681
6.525.2.8 getDestination . . . . .	2681
6.525.2.9 getMessage . . . . .	2681
6.525.2.10getMessage . . . . .	2681
6.525.2.11getRedeliveryCounter . . . . .	2681
6.525.2.12isMessageDispatch . . . . .	2681
6.525.2.13setConsumerId . . . . .	2682
6.525.2.14setDestination . . . . .	2682

6.525.2.15	setMessage	2682
6.525.2.16	setRedeliveryCounter	2682
6.525.2.17	toString	2682
6.525.2.18	visit	2682
6.525.3	Field Documentation	2683
6.525.3.1	consumerId	2683
6.525.3.2	destination	2683
6.525.3.3	ID_MESSAGE_DISPATCH	2683
6.525.3.4	message	2683
6.525.3.5	redeliveryCounter	2683
6.526	activemq::core::MessageDispatchChannel Class Reference	2683
6.526.1	Constructor & Destructor Documentation	2685
6.526.1.1	MessageDispatchChannel	2685
6.526.1.2	~MessageDispatchChannel	2685
6.526.2	Member Function Documentation	2685
6.526.2.1	clear	2685
6.526.2.2	close	2685
6.526.2.3	dequeue	2685
6.526.2.4	dequeueNoWait	2686
6.526.2.5	enqueue	2686
6.526.2.6	enqueueFirst	2686
6.526.2.7	isClosed	2686
6.526.2.8	isEmpty	2686
6.526.2.9	isRunning	2686
6.526.2.10	lock	2687
6.526.2.11	notify	2687
6.526.2.12	notifyAll	2687
6.526.2.13	peek	2688
6.526.2.14	removeAll	2688
6.526.2.15	size	2688
6.526.2.16	start	2688
6.526.2.17	stop	2688
6.526.2.18	tryLock	2688
6.526.2.19	unlock	2689
6.526.2.20	wait	2689
6.526.2.21	wait	2689
6.526.2.22	wait	2690
6.527	activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller	
	Class Reference	2690
6.527.1	Detailed Description	2691
6.527.2	Constructor & Destructor Documentation	2692
6.527.2.1	MessageDispatchMarshaller	2692
6.527.2.2	~MessageDispatchMarshaller	2692
6.527.3	Member Function Documentation	2692
6.527.3.1	createObject	2692
6.527.3.2	getDataStructureType	2692
6.527.3.3	looseMarshal	2692
6.527.3.4	looseUnmarshal	2693
6.527.3.5	tightMarshal1	2693
6.527.3.6	tightMarshal2	2694

6.527.3.7 tightUnmarshal . . . . .	2694
6.528activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	
Class Reference . . . . .	2695
6.528.1 Detailed Description . . . . .	2696
6.528.2 Constructor & Destructor Documentation . . . . .	2696
6.528.2.1 MessageDispatchMarshaller . . . . .	2696
6.528.2.2 ~MessageDispatchMarshaller . . . . .	2696
6.528.3 Member Function Documentation . . . . .	2696
6.528.3.1 createObject . . . . .	2696
6.528.3.2 getDataStructureType . . . . .	2696
6.528.3.3 looseMarshal . . . . .	2696
6.528.3.4 looseUnmarshal . . . . .	2697
6.528.3.5 tightMarshal1 . . . . .	2697
6.528.3.6 tightMarshal2 . . . . .	2698
6.528.3.7 tightUnmarshal . . . . .	2698
6.529activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller	
Class Reference . . . . .	2699
6.529.1 Detailed Description . . . . .	2700
6.529.2 Constructor & Destructor Documentation . . . . .	2700
6.529.2.1 MessageDispatchMarshaller . . . . .	2700
6.529.2.2 ~MessageDispatchMarshaller . . . . .	2700
6.529.3 Member Function Documentation . . . . .	2700
6.529.3.1 createObject . . . . .	2700
6.529.3.2 getDataStructureType . . . . .	2700
6.529.3.3 looseMarshal . . . . .	2701
6.529.3.4 looseUnmarshal . . . . .	2701
6.529.3.5 tightMarshal1 . . . . .	2702
6.529.3.6 tightMarshal2 . . . . .	2702
6.529.3.7 tightUnmarshal . . . . .	2703
6.530activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	
Class Reference . . . . .	2703
6.530.1 Detailed Description . . . . .	2704
6.530.2 Constructor & Destructor Documentation . . . . .	2704
6.530.2.1 MessageDispatchMarshaller . . . . .	2704
6.530.2.2 ~MessageDispatchMarshaller . . . . .	2704
6.530.3 Member Function Documentation . . . . .	2704
6.530.3.1 createObject . . . . .	2704
6.530.3.2 getDataStructureType . . . . .	2705
6.530.3.3 looseMarshal . . . . .	2705
6.530.3.4 looseUnmarshal . . . . .	2705
6.530.3.5 tightMarshal1 . . . . .	2706
6.530.3.6 tightMarshal2 . . . . .	2706
6.530.3.7 tightUnmarshal . . . . .	2707
6.531activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller	
Class Reference . . . . .	2707
6.531.1 Detailed Description . . . . .	2708
6.531.2 Constructor & Destructor Documentation . . . . .	2709
6.531.2.1 MessageDispatchMarshaller . . . . .	2709
6.531.2.2 ~MessageDispatchMarshaller . . . . .	2709
6.531.3 Member Function Documentation . . . . .	2709

6.531.3.1	createObject	2709
6.531.3.2	getDataStructureType	2709
6.531.3.3	looseMarshal	2709
6.531.3.4	looseUnmarshal	2710
6.531.3.5	tightMarshal1	2710
6.531.3.6	tightMarshal2	2711
6.531.3.7	tightUnmarshal	2711
6.532	activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller	
	Class Reference	2712
6.532.1	Detailed Description	2713
6.532.2	Constructor & Destructor Documentation	2713
6.532.2.1	MessageDispatchMarshaller	2713
6.532.2.2	~MessageDispatchMarshaller	2713
6.532.3	Member Function Documentation	2713
6.532.3.1	createObject	2713
6.532.3.2	getDataStructureType	2713
6.532.3.3	looseMarshal	2713
6.532.3.4	looseUnmarshal	2714
6.532.3.5	tightMarshal1	2714
6.532.3.6	tightMarshal2	2715
6.532.3.7	tightUnmarshal	2715
6.533	activemq::commands::MessageDispatchNotification Class Reference	2716
6.533.1	Constructor & Destructor Documentation	2717
6.533.1.1	MessageDispatchNotification	2717
6.533.1.2	~MessageDispatchNotification	2717
6.533.2	Member Function Documentation	2717
6.533.2.1	cloneDataStructure	2717
6.533.2.2	copyDataStructure	2718
6.533.2.3	equals	2718
6.533.2.4	getConsumerId	2718
6.533.2.5	getConsumerId	2718
6.533.2.6	getDataStructureType	2718
6.533.2.7	getDeliverySequenceId	2719
6.533.2.8	getDestination	2719
6.533.2.9	getDestination	2719
6.533.2.10	getMessageId	2719
6.533.2.11	getMessageId	2719
6.533.2.12	isMessageDispatchNotification	2719
6.533.2.13	setConsumerId	2719
6.533.2.14	setDeliverySequenceId	2719
6.533.2.15	setDestination	2719
6.533.2.16	setMessageId	2719
6.533.2.17	toString	2719
6.533.2.18	visit	2720
6.533.3	Field Documentation	2720
6.533.3.1	consumerId	2720
6.533.3.2	deliverySequenceId	2720
6.533.3.3	destination	2720
6.533.3.4	ID_MESSAGE_DISPATCH_NOTIFICATION	2720
6.533.3.5	messageId	2720

6.534activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller	
Class Reference . . . . .	2720
6.534.1 Detailed Description . . . . .	2721
6.534.2 Constructor & Destructor Documentation . . . . .	2722
6.534.2.1 MessageDispatchNotificationMarshaller . . . . .	2722
6.534.2.2 ~MessageDispatchNotificationMarshaller . . . . .	2722
6.534.3 Member Function Documentation . . . . .	2722
6.534.3.1 createObject . . . . .	2722
6.534.3.2 getDataStructureType . . . . .	2722
6.534.3.3 looseMarshal . . . . .	2722
6.534.3.4 looseUnmarshal . . . . .	2723
6.534.3.5 tightMarshal1 . . . . .	2723
6.534.3.6 tightMarshal2 . . . . .	2724
6.534.3.7 tightUnmarshal . . . . .	2724
6.535activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller	
Class Reference . . . . .	2725
6.535.1 Detailed Description . . . . .	2726
6.535.2 Constructor & Destructor Documentation . . . . .	2726
6.535.2.1 MessageDispatchNotificationMarshaller . . . . .	2726
6.535.2.2 ~MessageDispatchNotificationMarshaller . . . . .	2726
6.535.3 Member Function Documentation . . . . .	2726
6.535.3.1 createObject . . . . .	2726
6.535.3.2 getDataStructureType . . . . .	2726
6.535.3.3 looseMarshal . . . . .	2726
6.535.3.4 looseUnmarshal . . . . .	2727
6.535.3.5 tightMarshal1 . . . . .	2727
6.535.3.6 tightMarshal2 . . . . .	2728
6.535.3.7 tightUnmarshal . . . . .	2728
6.536activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller	
Class Reference . . . . .	2729
6.536.1 Detailed Description . . . . .	2730
6.536.2 Constructor & Destructor Documentation . . . . .	2730
6.536.2.1 MessageDispatchNotificationMarshaller . . . . .	2730
6.536.2.2 ~MessageDispatchNotificationMarshaller . . . . .	2730
6.536.3 Member Function Documentation . . . . .	2730
6.536.3.1 createObject . . . . .	2730
6.536.3.2 getDataStructureType . . . . .	2730
6.536.3.3 looseMarshal . . . . .	2731
6.536.3.4 looseUnmarshal . . . . .	2731
6.536.3.5 tightMarshal1 . . . . .	2732
6.536.3.6 tightMarshal2 . . . . .	2732
6.536.3.7 tightUnmarshal . . . . .	2733
6.537activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller	
Class Reference . . . . .	2733
6.537.1 Detailed Description . . . . .	2734
6.537.2 Constructor & Destructor Documentation . . . . .	2734
6.537.2.1 MessageDispatchNotificationMarshaller . . . . .	2734
6.537.2.2 ~MessageDispatchNotificationMarshaller . . . . .	2734
6.537.3 Member Function Documentation . . . . .	2734
6.537.3.1 createObject . . . . .	2734



6.537.3.2	getDataStructureType . . . . .	2735
6.537.3.3	looseMarshal . . . . .	2735
6.537.3.4	looseUnmarshal . . . . .	2735
6.537.3.5	tightMarshal1 . . . . .	2736
6.537.3.6	tightMarshal2 . . . . .	2736
6.537.3.7	tightUnmarshal . . . . .	2737
6.538	activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller	
	Class Reference . . . . .	2737
6.538.1	Detailed Description . . . . .	2738
6.538.2	Constructor & Destructor Documentation . . . . .	2739
6.538.2.1	MessageDispatchNotificationMarshaller . . . . .	2739
6.538.2.2	~MessageDispatchNotificationMarshaller . . . . .	2739
6.538.3	Member Function Documentation . . . . .	2739
6.538.3.1	createObject . . . . .	2739
6.538.3.2	getDataStructureType . . . . .	2739
6.538.3.3	looseMarshal . . . . .	2739
6.538.3.4	looseUnmarshal . . . . .	2740
6.538.3.5	tightMarshal1 . . . . .	2740
6.538.3.6	tightMarshal2 . . . . .	2741
6.538.3.7	tightUnmarshal . . . . .	2741
6.539	activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller	
	Class Reference . . . . .	2742
6.539.1	Detailed Description . . . . .	2743
6.539.2	Constructor & Destructor Documentation . . . . .	2743
6.539.2.1	MessageDispatchNotificationMarshaller . . . . .	2743
6.539.2.2	~MessageDispatchNotificationMarshaller . . . . .	2743
6.539.3	Member Function Documentation . . . . .	2743
6.539.3.1	createObject . . . . .	2743
6.539.3.2	getDataStructureType . . . . .	2743
6.539.3.3	looseMarshal . . . . .	2743
6.539.3.4	looseUnmarshal . . . . .	2744
6.539.3.5	tightMarshal1 . . . . .	2744
6.539.3.6	tightMarshal2 . . . . .	2745
6.539.3.7	tightUnmarshal . . . . .	2745
6.540	cms::MessageEnumeration Class Reference . . . . .	2746
6.540.1	Detailed Description . . . . .	2746
6.540.2	Constructor & Destructor Documentation . . . . .	2747
6.540.2.1	~MessageEnumeration . . . . .	2747
6.540.3	Member Function Documentation . . . . .	2747
6.540.3.1	hasMoreMessages . . . . .	2747
6.540.3.2	nextMessage . . . . .	2747
6.541	cms::MessageEOFException Class Reference . . . . .	2747
6.541.1	Detailed Description . . . . .	2748
6.541.2	Constructor & Destructor Documentation . . . . .	2748
6.541.2.1	MessageEOFException . . . . .	2748
6.541.2.2	MessageEOFException . . . . .	2748
6.541.2.3	MessageEOFException . . . . .	2748
6.541.2.4	MessageEOFException . . . . .	2748
6.541.2.5	~MessageEOFException . . . . .	2748
6.542	cms::MessageFormatException Class Reference . . . . .	2749

6.542.1 Detailed Description . . . . .	2749
6.542.2 Constructor & Destructor Documentation . . . . .	2750
6.542.2.1 MessageFormatException . . . . .	2750
6.542.2.2 MessageFormatException . . . . .	2750
6.542.2.3 MessageFormatException . . . . .	2750
6.542.2.4 MessageFormatException . . . . .	2750
6.542.2.5 ~MessageFormatException . . . . .	2750
6.543activemq::commands::MessageId Class Reference . . . . .	2750
6.543.1 Member Typedef Documentation . . . . .	2752
6.543.1.1 COMPARATOR . . . . .	2752
6.543.2 Constructor & Destructor Documentation . . . . .	2752
6.543.2.1 MessageId . . . . .	2752
6.543.2.2 MessageId . . . . .	2752
6.543.2.3 MessageId . . . . .	2752
6.543.2.4 MessageId . . . . .	2752
6.543.2.5 MessageId . . . . .	2752
6.543.2.6 MessageId . . . . .	2752
6.543.2.7 ~MessageId . . . . .	2752
6.543.3 Member Function Documentation . . . . .	2752
6.543.3.1 cloneDataStructure . . . . .	2752
6.543.3.2 compareTo . . . . .	2752
6.543.3.3 copyDataStructure . . . . .	2752
6.543.3.4 equals . . . . .	2753
6.543.3.5 equals . . . . .	2753
6.543.3.6 getBrokerSequenceId . . . . .	2753
6.543.3.7 getDataStructureType . . . . .	2753
6.543.3.8 getProducerId . . . . .	2754
6.543.3.9 getProducerId . . . . .	2754
6.543.3.10getProducerSequenceId . . . . .	2754
6.543.3.11operator< . . . . .	2754
6.543.3.12operator= . . . . .	2754
6.543.3.13operator== . . . . .	2754
6.543.3.14setBrokerSequenceId . . . . .	2754
6.543.3.15setProducerId . . . . .	2754
6.543.3.16setProducerSequenceId . . . . .	2754
6.543.3.17setTextView . . . . .	2754
6.543.3.18setValue . . . . .	2754
6.543.3.19toString . . . . .	2754
6.543.4 Field Documentation . . . . .	2755
6.543.4.1 brokerSequenceId . . . . .	2755
6.543.4.2 ID_MESSAGEID . . . . .	2755
6.543.4.3 producerId . . . . .	2755
6.543.4.4 producerSequenceId . . . . .	2755
6.544activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller Class	
Reference . . . . .	2755
6.544.1 Detailed Description . . . . .	2756
6.544.2 Constructor & Destructor Documentation . . . . .	2756
6.544.2.1 MessageIdMarshaller . . . . .	2756
6.544.2.2 ~MessageIdMarshaller . . . . .	2756
6.544.3 Member Function Documentation . . . . .	2756

6.544.3.1	createObject . . . . .	2756
6.544.3.2	getDataStructureType . . . . .	2757
6.544.3.3	looseMarshal . . . . .	2757
6.544.3.4	looseUnmarshal . . . . .	2757
6.544.3.5	tightMarshal1 . . . . .	2758
6.544.3.6	tightMarshal2 . . . . .	2758
6.544.3.7	tightUnmarshal . . . . .	2759
6.545	activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller Class	
	Reference . . . . .	2759
6.545.1	Detailed Description . . . . .	2760
6.545.2	Constructor & Destructor Documentation . . . . .	2760
6.545.2.1	MessageIdMarshaller . . . . .	2760
6.545.2.2	~MessageIdMarshaller . . . . .	2760
6.545.3	Member Function Documentation . . . . .	2760
6.545.3.1	createObject . . . . .	2760
6.545.3.2	getDataStructureType . . . . .	2761
6.545.3.3	looseMarshal . . . . .	2761
6.545.3.4	looseUnmarshal . . . . .	2761
6.545.3.5	tightMarshal1 . . . . .	2762
6.545.3.6	tightMarshal2 . . . . .	2762
6.545.3.7	tightUnmarshal . . . . .	2763
6.546	activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller Class	
	Reference . . . . .	2763
6.546.1	Detailed Description . . . . .	2764
6.546.2	Constructor & Destructor Documentation . . . . .	2764
6.546.2.1	MessageIdMarshaller . . . . .	2764
6.546.2.2	~MessageIdMarshaller . . . . .	2764
6.546.3	Member Function Documentation . . . . .	2764
6.546.3.1	createObject . . . . .	2764
6.546.3.2	getDataStructureType . . . . .	2765
6.546.3.3	looseMarshal . . . . .	2765
6.546.3.4	looseUnmarshal . . . . .	2765
6.546.3.5	tightMarshal1 . . . . .	2766
6.546.3.6	tightMarshal2 . . . . .	2766
6.546.3.7	tightUnmarshal . . . . .	2767
6.547	activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller Class	
	Reference . . . . .	2767
6.547.1	Detailed Description . . . . .	2768
6.547.2	Constructor & Destructor Documentation . . . . .	2768
6.547.2.1	MessageIdMarshaller . . . . .	2768
6.547.2.2	~MessageIdMarshaller . . . . .	2768
6.547.3	Member Function Documentation . . . . .	2768
6.547.3.1	createObject . . . . .	2768
6.547.3.2	getDataStructureType . . . . .	2769
6.547.3.3	looseMarshal . . . . .	2769
6.547.3.4	looseUnmarshal . . . . .	2769
6.547.3.5	tightMarshal1 . . . . .	2770
6.547.3.6	tightMarshal2 . . . . .	2770
6.547.3.7	tightUnmarshal . . . . .	2771

6.548activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller Class	
Reference . . . . .	2771
6.548.1 Detailed Description . . . . .	2772
6.548.2 Constructor & Destructor Documentation . . . . .	2772
6.548.2.1 MessageIdMarshaller . . . . .	2772
6.548.2.2 ~MessageIdMarshaller . . . . .	2772
6.548.3 Member Function Documentation . . . . .	2772
6.548.3.1 createObject . . . . .	2772
6.548.3.2 getDataStructureType . . . . .	2773
6.548.3.3 looseMarshal . . . . .	2773
6.548.3.4 looseUnmarshal . . . . .	2773
6.548.3.5 tightMarshal1 . . . . .	2774
6.548.3.6 tightMarshal2 . . . . .	2774
6.548.3.7 tightUnmarshal . . . . .	2775
6.549activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller Class	
Reference . . . . .	2775
6.549.1 Detailed Description . . . . .	2776
6.549.2 Constructor & Destructor Documentation . . . . .	2776
6.549.2.1 MessageIdMarshaller . . . . .	2776
6.549.2.2 ~MessageIdMarshaller . . . . .	2776
6.549.3 Member Function Documentation . . . . .	2776
6.549.3.1 createObject . . . . .	2776
6.549.3.2 getDataStructureType . . . . .	2777
6.549.3.3 looseMarshal . . . . .	2777
6.549.3.4 looseUnmarshal . . . . .	2777
6.549.3.5 tightMarshal1 . . . . .	2778
6.549.3.6 tightMarshal2 . . . . .	2778
6.549.3.7 tightUnmarshal . . . . .	2779
6.550cms::MessageListener Class Reference . . . . .	2779
6.550.1 Detailed Description . . . . .	2779
6.550.2 Constructor & Destructor Documentation . . . . .	2780
6.550.2.1 ~MessageListener . . . . .	2780
6.550.3 Member Function Documentation . . . . .	2780
6.550.3.1 onMessage . . . . .	2780
6.551activemq::wireformat::openwire::marshal::v5::MessageMarshaller Class	
Reference . . . . .	2780
6.551.1 Detailed Description . . . . .	2781
6.551.2 Constructor & Destructor Documentation . . . . .	2781
6.551.2.1 MessageMarshaller . . . . .	2781
6.551.2.2 ~MessageMarshaller . . . . .	2781
6.551.3 Member Function Documentation . . . . .	2781
6.551.3.1 looseMarshal . . . . .	2781
6.551.3.2 looseUnmarshal . . . . .	2782
6.551.3.3 tightMarshal1 . . . . .	2783
6.551.3.4 tightMarshal2 . . . . .	2783
6.551.3.5 tightUnmarshal . . . . .	2784
6.552activemq::wireformat::openwire::marshal::v3::MessageMarshaller Class	
Reference . . . . .	2785
6.552.1 Detailed Description . . . . .	2785
6.552.2 Constructor & Destructor Documentation . . . . .	2786

6.552.2.1 MessageMarshaller . . . . .	2786
6.552.2.2 ~MessageMarshaller . . . . .	2786
6.552.3 Member Function Documentation . . . . .	2786
6.552.3.1 looseMarshal . . . . .	2786
6.552.3.2 looseUnmarshal . . . . .	2786
6.552.3.3 tightMarshal1 . . . . .	2787
6.552.3.4 tightMarshal2 . . . . .	2788
6.552.3.5 tightUnmarshal . . . . .	2788
6.553activemq::wireformat::openwire::marshal::v2::MessageMarshaller Class	
Reference . . . . .	2789
6.553.1 Detailed Description . . . . .	2790
6.553.2 Constructor & Destructor Documentation . . . . .	2790
6.553.2.1 MessageMarshaller . . . . .	2790
6.553.2.2 ~MessageMarshaller . . . . .	2790
6.553.3 Member Function Documentation . . . . .	2790
6.553.3.1 looseMarshal . . . . .	2790
6.553.3.2 looseUnmarshal . . . . .	2791
6.553.3.3 tightMarshal1 . . . . .	2791
6.553.3.4 tightMarshal2 . . . . .	2792
6.553.3.5 tightUnmarshal . . . . .	2793
6.554activemq::wireformat::openwire::marshal::v4::MessageMarshaller Class	
Reference . . . . .	2793
6.554.1 Detailed Description . . . . .	2794
6.554.2 Constructor & Destructor Documentation . . . . .	2795
6.554.2.1 MessageMarshaller . . . . .	2795
6.554.2.2 ~MessageMarshaller . . . . .	2795
6.554.3 Member Function Documentation . . . . .	2795
6.554.3.1 looseMarshal . . . . .	2795
6.554.3.2 looseUnmarshal . . . . .	2795
6.554.3.3 tightMarshal1 . . . . .	2796
6.554.3.4 tightMarshal2 . . . . .	2797
6.554.3.5 tightUnmarshal . . . . .	2797
6.555activemq::wireformat::openwire::marshal::v1::MessageMarshaller Class	
Reference . . . . .	2798
6.555.1 Detailed Description . . . . .	2799
6.555.2 Constructor & Destructor Documentation . . . . .	2799
6.555.2.1 MessageMarshaller . . . . .	2799
6.555.2.2 ~MessageMarshaller . . . . .	2799
6.555.3 Member Function Documentation . . . . .	2799
6.555.3.1 looseMarshal . . . . .	2799
6.555.3.2 looseUnmarshal . . . . .	2800
6.555.3.3 tightMarshal1 . . . . .	2800
6.555.3.4 tightMarshal2 . . . . .	2801
6.555.3.5 tightUnmarshal . . . . .	2802
6.556activemq::wireformat::openwire::marshal::v6::MessageMarshaller Class	
Reference . . . . .	2802
6.556.1 Detailed Description . . . . .	2803
6.556.2 Constructor & Destructor Documentation . . . . .	2804
6.556.2.1 MessageMarshaller . . . . .	2804
6.556.2.2 ~MessageMarshaller . . . . .	2804

6.556.3 Member Function Documentation . . . . .	2804
6.556.3.1 looseMarshal . . . . .	2804
6.556.3.2 looseUnmarshal . . . . .	2804
6.556.3.3 tightMarshal1 . . . . .	2805
6.556.3.4 tightMarshal2 . . . . .	2806
6.556.3.5 tightUnmarshal . . . . .	2806
6.557cms::MessageNotReadableException Class Reference . . . . .	2807
6.557.1 Detailed Description . . . . .	2807
6.557.2 Constructor & Destructor Documentation . . . . .	2808
6.557.2.1 MessageNotReadableException . . . . .	2808
6.557.2.2 MessageNotReadableException . . . . .	2808
6.557.2.3 MessageNotReadableException . . . . .	2808
6.557.2.4 MessageNotReadableException . . . . .	2808
6.557.2.5 ~MessageNotReadableException . . . . .	2808
6.558cms::MessageNotWriteableException Class Reference . . . . .	2808
6.558.1 Detailed Description . . . . .	2808
6.558.2 Constructor & Destructor Documentation . . . . .	2809
6.558.2.1 MessageNotWriteableException . . . . .	2809
6.558.2.2 MessageNotWriteableException . . . . .	2809
6.558.2.3 MessageNotWriteableException . . . . .	2809
6.558.2.4 MessageNotWriteableException . . . . .	2809
6.558.2.5 ~MessageNotWriteableException . . . . .	2809
6.559cms::MessageProducer Class Reference . . . . .	2809
6.559.1 Detailed Description . . . . .	2811
6.559.2 Constructor & Destructor Documentation . . . . .	2811
6.559.2.1 ~MessageProducer . . . . .	2811
6.559.3 Member Function Documentation . . . . .	2811
6.559.3.1 getDeliveryMode . . . . .	2811
6.559.3.2 getDisableMessageID . . . . .	2812
6.559.3.3 getDisableMessageTimeStamp . . . . .	2812
6.559.3.4 getPriority . . . . .	2812
6.559.3.5 getTimeToLive . . . . .	2813
6.559.3.6 send . . . . .	2813
6.559.3.7 send . . . . .	2814
6.559.3.8 send . . . . .	2814
6.559.3.9 send . . . . .	2815
6.559.3.10setDeliveryMode . . . . .	2816
6.559.3.11setDisableMessageID . . . . .	2816
6.559.3.12setDisableMessageTimeStamp . . . . .	2816
6.559.3.13setPriority . . . . .	2817
6.559.3.14setTimeToLive . . . . .	2817
6.560activemq::wireformat::openwire::utils::MessagePropertyInterceptor Class Reference . . . . .	2817
6.560.1 Detailed Description . . . . .	2819
6.560.2 Constructor & Destructor Documentation . . . . .	2819
6.560.2.1 MessagePropertyInterceptor . . . . .	2819
6.560.2.2 ~MessagePropertyInterceptor . . . . .	2820
6.560.3 Member Function Documentation . . . . .	2820
6.560.3.1 getBooleanProperty . . . . .	2820
6.560.3.2 getByteProperty . . . . .	2820

6.560.3.3	getDoubleProperty . . . . .	2820
6.560.3.4	getFloatProperty . . . . .	2820
6.560.3.5	getIntProperty . . . . .	2821
6.560.3.6	getLongProperty . . . . .	2821
6.560.3.7	getShortProperty . . . . .	2821
6.560.3.8	getStringProperty . . . . .	2822
6.560.3.9	setBooleanProperty . . . . .	2822
6.560.3.10	setByteProperty . . . . .	2822
6.560.3.11	setDoubleProperty . . . . .	2822
6.560.3.12	setFloatProperty . . . . .	2823
6.560.3.13	setIntProperty . . . . .	2823
6.560.3.14	setLongProperty . . . . .	2823
6.560.3.15	setShortProperty . . . . .	2823
6.560.3.16	setStringProperty . . . . .	2823
6.561	activemq::commands::MessagePull Class Reference . . . . .	2824
6.561.1	Constructor & Destructor Documentation . . . . .	2825
6.561.1.1	MessagePull . . . . .	2825
6.561.1.2	~MessagePull . . . . .	2825
6.561.2	Member Function Documentation . . . . .	2825
6.561.2.1	cloneDataStructure . . . . .	2825
6.561.2.2	copyDataStructure . . . . .	2826
6.561.2.3	equals . . . . .	2826
6.561.2.4	getConsumerId . . . . .	2826
6.561.2.5	getConsumerId . . . . .	2826
6.561.2.6	getCorrelationId . . . . .	2826
6.561.2.7	getCorrelationId . . . . .	2826
6.561.2.8	getDataStructureType . . . . .	2826
6.561.2.9	getDestination . . . . .	2827
6.561.2.10	getDestination . . . . .	2827
6.561.2.11	getMessageId . . . . .	2827
6.561.2.12	getMessageId . . . . .	2827
6.561.2.13	getTimeout . . . . .	2827
6.561.2.14	setConsumerId . . . . .	2827
6.561.2.15	setCorrelationId . . . . .	2827
6.561.2.16	setDestination . . . . .	2827
6.561.2.17	setMessageId . . . . .	2827
6.561.2.18	setTimeout . . . . .	2827
6.561.2.19	toString . . . . .	2827
6.561.2.20	visit . . . . .	2828
6.561.3	Field Documentation . . . . .	2828
6.561.3.1	consumerId . . . . .	2828
6.561.3.2	correlationId . . . . .	2828
6.561.3.3	destination . . . . .	2828
6.561.3.4	ID_MESSAGEPULL . . . . .	2828
6.561.3.5	messageId . . . . .	2828
6.561.3.6	timeout . . . . .	2828
6.562	activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller	
	Class Reference . . . . .	2828
6.562.1	Detailed Description . . . . .	2829
6.562.2	Constructor & Destructor Documentation . . . . .	2830

6.562.2.1 MessagePullMarshaller . . . . .	2830
6.562.2.2 ~MessagePullMarshaller . . . . .	2830
6.562.3 Member Function Documentation . . . . .	2830
6.562.3.1 createObject . . . . .	2830
6.562.3.2 getDataStructureType . . . . .	2830
6.562.3.3 looseMarshal . . . . .	2830
6.562.3.4 looseUnmarshal . . . . .	2831
6.562.3.5 tightMarshal1 . . . . .	2831
6.562.3.6 tightMarshal2 . . . . .	2832
6.562.3.7 tightUnmarshal . . . . .	2832
6.563activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller	
Class Reference . . . . .	2833
6.563.1 Detailed Description . . . . .	2834
6.563.2 Constructor & Destructor Documentation . . . . .	2834
6.563.2.1 MessagePullMarshaller . . . . .	2834
6.563.2.2 ~MessagePullMarshaller . . . . .	2834
6.563.3 Member Function Documentation . . . . .	2834
6.563.3.1 createObject . . . . .	2834
6.563.3.2 getDataStructureType . . . . .	2834
6.563.3.3 looseMarshal . . . . .	2834
6.563.3.4 looseUnmarshal . . . . .	2835
6.563.3.5 tightMarshal1 . . . . .	2835
6.563.3.6 tightMarshal2 . . . . .	2836
6.563.3.7 tightUnmarshal . . . . .	2836
6.564activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller	
Class Reference . . . . .	2837
6.564.1 Detailed Description . . . . .	2838
6.564.2 Constructor & Destructor Documentation . . . . .	2838
6.564.2.1 MessagePullMarshaller . . . . .	2838
6.564.2.2 ~MessagePullMarshaller . . . . .	2838
6.564.3 Member Function Documentation . . . . .	2838
6.564.3.1 createObject . . . . .	2838
6.564.3.2 getDataStructureType . . . . .	2838
6.564.3.3 looseMarshal . . . . .	2839
6.564.3.4 looseUnmarshal . . . . .	2839
6.564.3.5 tightMarshal1 . . . . .	2840
6.564.3.6 tightMarshal2 . . . . .	2840
6.564.3.7 tightUnmarshal . . . . .	2841
6.565activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	
Class Reference . . . . .	2841
6.565.1 Detailed Description . . . . .	2842
6.565.2 Constructor & Destructor Documentation . . . . .	2842
6.565.2.1 MessagePullMarshaller . . . . .	2842
6.565.2.2 ~MessagePullMarshaller . . . . .	2842
6.565.3 Member Function Documentation . . . . .	2842
6.565.3.1 createObject . . . . .	2842
6.565.3.2 getDataStructureType . . . . .	2843
6.565.3.3 looseMarshal . . . . .	2843
6.565.3.4 looseUnmarshal . . . . .	2843
6.565.3.5 tightMarshal1 . . . . .	2844



6.565.3.6	tightMarshal2	2844
6.565.3.7	tightUnmarshal	2845
6.566	activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller	
	Class Reference	2845
6.566.1	Detailed Description	2846
6.566.2	Constructor & Destructor Documentation	2847
6.566.2.1	MessagePullMarshaller	2847
6.566.2.2	~MessagePullMarshaller	2847
6.566.3	Member Function Documentation	2847
6.566.3.1	createObject	2847
6.566.3.2	getDataStructureType	2847
6.566.3.3	looseMarshal	2847
6.566.3.4	looseUnmarshal	2848
6.566.3.5	tightMarshal1	2848
6.566.3.6	tightMarshal2	2849
6.566.3.7	tightUnmarshal	2849
6.567	activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller	
	Class Reference	2850
6.567.1	Detailed Description	2851
6.567.2	Constructor & Destructor Documentation	2851
6.567.2.1	MessagePullMarshaller	2851
6.567.2.2	~MessagePullMarshaller	2851
6.567.3	Member Function Documentation	2851
6.567.3.1	createObject	2851
6.567.3.2	getDataStructureType	2851
6.567.3.3	looseMarshal	2851
6.567.3.4	looseUnmarshal	2852
6.567.3.5	tightMarshal1	2852
6.567.3.6	tightMarshal2	2853
6.567.3.7	tightUnmarshal	2853
6.568	activemq::transport::mock::MockTransport Class Reference	2854
6.568.1	Detailed Description	2856
6.568.2	Constructor & Destructor Documentation	2857
6.568.2.1	MockTransport	2857
6.568.2.2	~MockTransport	2857
6.568.3	Member Function Documentation	2857
6.568.3.1	close	2857
6.568.3.2	fireCommand	2857
6.568.3.3	fireException	2857
6.568.3.4	getInstance	2858
6.568.3.5	getNumReceivedMessageBeforeFail	2858
6.568.3.6	getNumReceivedMessages	2858
6.568.3.7	getNumSentKeepAlives	2858
6.568.3.8	getNumSentKeepAlivesBeforeFail	2858
6.568.3.9	getNumSentMessageBeforeFail	2858
6.568.3.10	getNumSentMessages	2858
6.568.3.11	getRemoteAddress	2858
6.568.3.12	getTransportListener	2858
6.568.3.13	getWireFormat	2858
6.568.3.14	isClosed	2859

6.568.3.15	isConnected . . . . .	2859
6.568.3.16	isFailOnClose . . . . .	2859
6.568.3.17	isFailOnKeepAliveSends . . . . .	2859
6.568.3.18	isFailOnReceiveMessage . . . . .	2859
6.568.3.19	isFailOnSendMessage . . . . .	2859
6.568.3.20	isFailOnStart . . . . .	2859
6.568.3.21	isFailOnStop . . . . .	2859
6.568.3.22	isFaultTolerant . . . . .	2859
6.568.3.23	narrow . . . . .	2860
6.568.3.24	oneway . . . . .	2860
6.568.3.25	reconnect . . . . .	2860
6.568.3.26	request . . . . .	2861
6.568.3.27	request . . . . .	2861
6.568.3.28	setFailOnClose . . . . .	2862
6.568.3.29	setFailOnKeepAliveSends . . . . .	2862
6.568.3.30	setFailOnReceiveMessage . . . . .	2862
6.568.3.31	setFailOnSendMessage . . . . .	2862
6.568.3.32	setFailOnStart . . . . .	2862
6.568.3.33	setFailOnStop . . . . .	2862
6.568.3.34	setNumReceivedMessageBeforeFail . . . . .	2862
6.568.3.35	setNumReceivedMessages . . . . .	2862
6.568.3.36	setNumSentKeepAlives . . . . .	2862
6.568.3.37	setNumSentKeepAlivesBeforeFail . . . . .	2862
6.568.3.38	setNumSentMessageBeforeFail . . . . .	2862
6.568.3.39	setNumSentMessages . . . . .	2862
6.568.3.40	setOutgoingListener . . . . .	2862
6.568.3.41	setResponseBuilder . . . . .	2863
6.568.3.42	setTransportListener . . . . .	2863
6.568.3.43	setWireFormat . . . . .	2863
6.568.3.44	start . . . . .	2863
6.568.3.45	stop . . . . .	2864
6.569	activemq::transport::mock::MockTransportFactory Class Reference . . . . .	2864
6.569.1	Detailed Description . . . . .	2865
6.569.2	Constructor & Destructor Documentation . . . . .	2865
6.569.2.1	~MockTransportFactory . . . . .	2865
6.569.3	Member Function Documentation . . . . .	2865
6.569.3.1	create . . . . .	2865
6.569.3.2	createComposite . . . . .	2865
6.569.3.3	doCreateComposite . . . . .	2866
6.570	decaf::util::concurrent::Mutex Class Reference . . . . .	2866
6.570.1	Detailed Description . . . . .	2867
6.570.2	Constructor & Destructor Documentation . . . . .	2868
6.570.2.1	Mutex . . . . .	2868
6.570.2.2	~Mutex . . . . .	2868
6.570.3	Member Function Documentation . . . . .	2868
6.570.3.1	lock . . . . .	2868
6.570.3.2	notify . . . . .	2868
6.570.3.3	notifyAll . . . . .	2868
6.570.3.4	tryLock . . . . .	2869
6.570.3.5	unlock . . . . .	2869

6.570.3.6 wait . . . . .	2870
6.570.3.7 wait . . . . .	2870
6.570.3.8 wait . . . . .	2871
6.571decaf::util::concurrent::MutexHandle Class Reference . . . . .	2871
6.571.1 Constructor & Destructor Documentation . . . . .	2872
6.571.1.1 MutexHandle . . . . .	2872
6.571.1.2 ~MutexHandle . . . . .	2872
6.571.1.3 MutexHandle . . . . .	2872
6.571.1.4 ~MutexHandle . . . . .	2872
6.571.2 Field Documentation . . . . .	2872
6.571.2.1 lock_count . . . . .	2872
6.571.2.2 lock_owner . . . . .	2872
6.571.2.3 mutex . . . . .	2872
6.571.2.4 mutex . . . . .	2872
6.572decaf::internal::util::concurrent::MutexImpl Class Reference . . . . .	2872
6.572.1 Member Function Documentation . . . . .	2873
6.572.1.1 create . . . . .	2873
6.572.1.2 destroy . . . . .	2873
6.572.1.3 lock . . . . .	2873
6.572.1.4 trylock . . . . .	2874
6.572.1.5 unlock . . . . .	2874
6.573decaf::internal::net::Network Class Reference . . . . .	2874
6.573.1 Detailed Description . . . . .	2875
6.573.2 Constructor & Destructor Documentation . . . . .	2876
6.573.2.1 Network . . . . .	2876
6.573.2.2 ~Network . . . . .	2876
6.573.3 Member Function Documentation . . . . .	2876
6.573.3.1 addAsResource . . . . .	2876
6.573.3.2 addNetworkResource . . . . .	2876
6.573.3.3 getNetworkRuntime . . . . .	2876
6.573.3.4 getRuntimeLock . . . . .	2876
6.573.3.5 initializeNetworking . . . . .	2877
6.573.3.6 shutdownNetworking . . . . .	2877
6.574activemq::commands::NetworkBridgeFilter Class Reference . . . . .	2877
6.574.1 Constructor & Destructor Documentation . . . . .	2878
6.574.1.1 NetworkBridgeFilter . . . . .	2878
6.574.1.2 ~NetworkBridgeFilter . . . . .	2878
6.574.2 Member Function Documentation . . . . .	2878
6.574.2.1 cloneDataStructure . . . . .	2878
6.574.2.2 copyDataStructure . . . . .	2878
6.574.2.3 equals . . . . .	2879
6.574.2.4 getDataStructureType . . . . .	2879
6.574.2.5 getNetworkBrokerId . . . . .	2880
6.574.2.6 getNetworkBrokerId . . . . .	2880
6.574.2.7 getNetworkTTL . . . . .	2880
6.574.2.8 setNetworkBrokerId . . . . .	2880
6.574.2.9 setNetworkTTL . . . . .	2880
6.574.2.10toString . . . . .	2880
6.574.3 Field Documentation . . . . .	2880
6.574.3.1 ID_NETWORKBRIDGEFILTER . . . . .	2880

6.574.3.2 networkBrokerId . . . . .	2880
6.574.3.3 networkTTL . . . . .	2880
6.575activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller	
Class Reference . . . . .	2881
6.575.1 Detailed Description . . . . .	2882
6.575.2 Constructor & Destructor Documentation . . . . .	2882
6.575.2.1 NetworkBridgeFilterMarshaller . . . . .	2882
6.575.2.2 ~NetworkBridgeFilterMarshaller . . . . .	2882
6.575.3 Member Function Documentation . . . . .	2882
6.575.3.1 createObject . . . . .	2882
6.575.3.2 getDataStructureType . . . . .	2882
6.575.3.3 looseMarshal . . . . .	2882
6.575.3.4 looseUnmarshal . . . . .	2883
6.575.3.5 tightMarshal1 . . . . .	2883
6.575.3.6 tightMarshal2 . . . . .	2884
6.575.3.7 tightUnmarshal . . . . .	2884
6.576activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller	
Class Reference . . . . .	2885
6.576.1 Detailed Description . . . . .	2886
6.576.2 Constructor & Destructor Documentation . . . . .	2886
6.576.2.1 NetworkBridgeFilterMarshaller . . . . .	2886
6.576.2.2 ~NetworkBridgeFilterMarshaller . . . . .	2886
6.576.3 Member Function Documentation . . . . .	2886
6.576.3.1 createObject . . . . .	2886
6.576.3.2 getDataStructureType . . . . .	2886
6.576.3.3 looseMarshal . . . . .	2886
6.576.3.4 looseUnmarshal . . . . .	2887
6.576.3.5 tightMarshal1 . . . . .	2887
6.576.3.6 tightMarshal2 . . . . .	2888
6.576.3.7 tightUnmarshal . . . . .	2888
6.577activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller	
Class Reference . . . . .	2889
6.577.1 Detailed Description . . . . .	2890
6.577.2 Constructor & Destructor Documentation . . . . .	2890
6.577.2.1 NetworkBridgeFilterMarshaller . . . . .	2890
6.577.2.2 ~NetworkBridgeFilterMarshaller . . . . .	2890
6.577.3 Member Function Documentation . . . . .	2890
6.577.3.1 createObject . . . . .	2890
6.577.3.2 getDataStructureType . . . . .	2890
6.577.3.3 looseMarshal . . . . .	2890
6.577.3.4 looseUnmarshal . . . . .	2891
6.577.3.5 tightMarshal1 . . . . .	2891
6.577.3.6 tightMarshal2 . . . . .	2892
6.577.3.7 tightUnmarshal . . . . .	2892
6.578activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller	
Class Reference . . . . .	2893
6.578.1 Detailed Description . . . . .	2894
6.578.2 Constructor & Destructor Documentation . . . . .	2894
6.578.2.1 NetworkBridgeFilterMarshaller . . . . .	2894
6.578.2.2 ~NetworkBridgeFilterMarshaller . . . . .	2894

6.578.3 Member Function Documentation . . . . .	2894
6.578.3.1 createObject . . . . .	2894
6.578.3.2 getDataStructureType . . . . .	2894
6.578.3.3 looseMarshal . . . . .	2894
6.578.3.4 looseUnmarshal . . . . .	2895
6.578.3.5 tightMarshal1 . . . . .	2895
6.578.3.6 tightMarshal2 . . . . .	2896
6.578.3.7 tightUnmarshal . . . . .	2896
6.579activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller	
Class Reference . . . . .	2897
6.579.1 Detailed Description . . . . .	2898
6.579.2 Constructor & Destructor Documentation . . . . .	2898
6.579.2.1 NetworkBridgeFilterMarshaller . . . . .	2898
6.579.2.2 ~NetworkBridgeFilterMarshaller . . . . .	2898
6.579.3 Member Function Documentation . . . . .	2898
6.579.3.1 createObject . . . . .	2898
6.579.3.2 getDataStructureType . . . . .	2898
6.579.3.3 looseMarshal . . . . .	2898
6.579.3.4 looseUnmarshal . . . . .	2899
6.579.3.5 tightMarshal1 . . . . .	2899
6.579.3.6 tightMarshal2 . . . . .	2900
6.579.3.7 tightUnmarshal . . . . .	2900
6.580activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller	
Class Reference . . . . .	2901
6.580.1 Detailed Description . . . . .	2902
6.580.2 Constructor & Destructor Documentation . . . . .	2902
6.580.2.1 NetworkBridgeFilterMarshaller . . . . .	2902
6.580.2.2 ~NetworkBridgeFilterMarshaller . . . . .	2902
6.580.3 Member Function Documentation . . . . .	2902
6.580.3.1 createObject . . . . .	2902
6.580.3.2 getDataStructureType . . . . .	2902
6.580.3.3 looseMarshal . . . . .	2902
6.580.3.4 looseUnmarshal . . . . .	2903
6.580.3.5 tightMarshal1 . . . . .	2903
6.580.3.6 tightMarshal2 . . . . .	2904
6.580.3.7 tightUnmarshal . . . . .	2904
6.581decaf::net::NoRouteToHostException Class Reference . . . . .	2905
6.581.1 Constructor & Destructor Documentation . . . . .	2905
6.581.1.1 NoRouteToHostException . . . . .	2905
6.581.1.2 NoRouteToHostException . . . . .	2906
6.581.1.3 NoRouteToHostException . . . . .	2906
6.581.1.4 NoRouteToHostException . . . . .	2906
6.581.1.5 NoRouteToHostException . . . . .	2906
6.581.1.6 NoRouteToHostException . . . . .	2907
6.581.1.7 ~NoRouteToHostException . . . . .	2907
6.581.2 Member Function Documentation . . . . .	2907
6.581.2.1 clone . . . . .	2907
6.582decaf::security::NoSuchAlgorithmException Class Reference . . . . .	2907
6.582.1 Constructor & Destructor Documentation . . . . .	2908
6.582.1.1 NoSuchAlgorithmException . . . . .	2908

6.582.1.2 NoSuchAlgorithmException	2908
6.582.1.3 NoSuchAlgorithmException	2908
6.582.1.4 NoSuchAlgorithmException	2909
6.582.1.5 NoSuchAlgorithmException	2909
6.582.1.6 NoSuchAlgorithmException	2909
6.582.1.7 ~NoSuchAlgorithmException	2910
6.582.2 Member Function Documentation	2910
6.582.2.1 clone	2910
6.583decaf::lang::exceptions::NoSuchElementException Class Reference	2910
6.583.1 Constructor & Destructor Documentation	2911
6.583.1.1 NoSuchElementException	2911
6.583.1.2 NoSuchElementException	2911
6.583.1.3 NoSuchElementException	2911
6.583.1.4 NoSuchElementException	2911
6.583.1.5 NoSuchElementException	2912
6.583.1.6 NoSuchElementException	2912
6.583.1.7 ~NoSuchElementException	2912
6.583.2 Member Function Documentation	2912
6.583.2.1 clone	2912
6.584decaf::security::NoSuchProviderException Class Reference	2913
6.584.1 Constructor & Destructor Documentation	2913
6.584.1.1 NoSuchProviderException	2913
6.584.1.2 NoSuchProviderException	2914
6.584.1.3 NoSuchProviderException	2914
6.584.1.4 NoSuchProviderException	2914
6.584.1.5 NoSuchProviderException	2914
6.584.1.6 NoSuchProviderException	2915
6.584.1.7 ~NoSuchProviderException	2915
6.584.2 Member Function Documentation	2915
6.584.2.1 clone	2915
6.585decaf::lang::exceptions::NullPointerException Class Reference	2915
6.585.1 Constructor & Destructor Documentation	2916
6.585.1.1 NullPointerException	2916
6.585.1.2 NullPointerException	2916
6.585.1.3 NullPointerException	2916
6.585.1.4 NullPointerException	2917
6.585.1.5 NullPointerException	2917
6.585.1.6 NullPointerException	2917
6.585.1.7 ~NullPointerException	2918
6.585.2 Member Function Documentation	2918
6.585.2.1 clone	2918
6.586decaf::lang::Number Class Reference	2918
6.586.1 Detailed Description	2919
6.586.2 Constructor & Destructor Documentation	2919
6.586.2.1 ~Number	2919
6.586.3 Member Function Documentation	2919
6.586.3.1 byteValue	2919
6.586.3.2 doubleValue	2919
6.586.3.3 floatValue	2920
6.586.3.4 intValue	2920

6.586.3.5 longValue . . . . .	2920
6.586.3.6 shortValue . . . . .	2920
6.587decaf::lang::exceptions::NumberFormatException Class Reference . .	2921
6.587.1 Constructor & Destructor Documentation . . . . .	2922
6.587.1.1 NumberFormatException . . . . .	2922
6.587.1.2 NumberFormatException . . . . .	2922
6.587.1.3 NumberFormatException . . . . .	2922
6.587.1.4 NumberFormatException . . . . .	2922
6.587.1.5 NumberFormatException . . . . .	2923
6.587.1.6 NumberFormatException . . . . .	2923
6.587.1.7 ~NumberFormatException . . . . .	2923
6.587.2 Member Function Documentation . . . . .	2923
6.587.2.1 clone . . . . .	2923
6.588cms::ObjectMessage Class Reference . . . . .	2924
6.588.1 Detailed Description . . . . .	2924
6.588.2 Constructor & Destructor Documentation . . . . .	2924
6.588.2.1 ~ObjectMessage . . . . .	2924
6.589decaf::internal::net::ssl::openssl::OpenSSLContextSpi Class Reference	2924
6.589.1 Detailed Description . . . . .	2925
6.589.2 Constructor & Destructor Documentation . . . . .	2926
6.589.2.1 OpenSSLContextSpi . . . . .	2926
6.589.2.2 ~OpenSSLContextSpi . . . . .	2926
6.589.3 Member Function Documentation . . . . .	2926
6.589.3.1 providerGetServerSocketFactory . . . . .	2926
6.589.3.2 providerGetSocketFactory . . . . .	2926
6.589.3.3 providerInit . . . . .	2927
6.589.4 Friends And Related Function Documentation . . . . .	2927
6.589.4.1 OpenSSLSocket . . . . .	2927
6.589.4.2 OpenSSLSocketFactory . . . . .	2927
6.590decaf::internal::net::ssl::openssl::OpenSSLParameters Class Reference	2927
6.590.1 Detailed Description . . . . .	2928
6.590.2 Constructor & Destructor Documentation . . . . .	2928
6.590.2.1 ~OpenSSLParameters . . . . .	2928
6.590.3 Member Function Documentation . . . . .	2928
6.590.3.1 clone . . . . .	2928
6.590.3.2 getEnabledCipherSuites . . . . .	2929
6.590.3.3 getEnabledProtocols . . . . .	2929
6.590.3.4 getNeedClientAuth . . . . .	2929
6.590.3.5 getSupportedCipherSuites . . . . .	2929
6.590.3.6 getSupportedProtocols . . . . .	2929
6.590.3.7 getUseClientMode . . . . .	2929
6.590.3.8 getWantClientAuth . . . . .	2929
6.590.3.9 setEnabledCipherSuites . . . . .	2929
6.590.3.10setEnabledProtocols . . . . .	2929
6.590.3.11setNeedClientAuth . . . . .	2929
6.590.3.12setUseClientMode . . . . .	2929
6.590.3.13setWantClientAuth . . . . .	2929
6.591decaf::internal::net::ssl::openssl::OpenSSLServerSocket Class Reference	2929
6.591.1 Detailed Description . . . . .	2932
6.591.2 Constructor & Destructor Documentation . . . . .	2932

6.591.2.1	OpenSSLServerSocket	2932
6.591.2.2	~OpenSSLServerSocket	2932
6.591.3	Member Function Documentation	2932
6.591.3.1	accept	2932
6.591.3.2	getEnabledCipherSuites	2933
6.591.3.3	getEnabledProtocols	2933
6.591.3.4	getNeedClientAuth	2933
6.591.3.5	getSupportedCipherSuites	2933
6.591.3.6	getSupportedProtocols	2934
6.591.3.7	getWantClientAuth	2934
6.591.3.8	setEnabledCipherSuites	2934
6.591.3.9	setEnabledProtocols	2934
6.591.3.10	setNeedClientAuth	2935
6.591.3.11	setWantClientAuth	2935
6.592	decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory Class Reference	2935
6.592.1	Detailed Description	2937
6.592.2	Constructor & Destructor Documentation	2938
6.592.2.1	OpenSSLServerSocketFactory	2938
6.592.2.2	~OpenSSLServerSocketFactory	2938
6.592.3	Member Function Documentation	2938
6.592.3.1	createServerSocket	2938
6.592.3.2	createServerSocket	2938
6.592.3.3	createServerSocket	2939
6.592.3.4	createServerSocket	2939
6.592.3.5	getDefaultCipherSuites	2940
6.592.3.6	getSupportedCipherSuites	2940
6.593	decaf::internal::net::ssl::openssl::OpenSSLSocket Class Reference	2940
6.593.1	Detailed Description	2946
6.593.2	Constructor & Destructor Documentation	2946
6.593.2.1	OpenSSLSocket	2946
6.593.2.2	OpenSSLSocket	2946
6.593.2.3	OpenSSLSocket	2946
6.593.2.4	OpenSSLSocket	2946
6.593.2.5	OpenSSLSocket	2946
6.593.2.6	~OpenSSLSocket	2946
6.593.3	Member Function Documentation	2946
6.593.3.1	available	2946
6.593.3.2	close	2947
6.593.3.3	connect	2947
6.593.3.4	getEnabledCipherSuites	2947
6.593.3.5	getEnabledProtocols	2948
6.593.3.6	getInputStream	2948
6.593.3.7	getNeedClientAuth	2948
6.593.3.8	getOutputStream	2949
6.593.3.9	getSupportedCipherSuites	2949
6.593.3.10	getSupportedProtocols	2949
6.593.3.11	getUseClientMode	2950
6.593.3.12	getWantClientAuth	2950
6.593.3.13	read	2950



6.593.3.14	sendUrgentData . . . . .	2951
6.593.3.15	setEnabledCipherSuites . . . . .	2951
6.593.3.16	setEnabledProtocols . . . . .	2951
6.593.3.17	setNeedClientAuth . . . . .	2952
6.593.3.18	setOOBInline . . . . .	2952
6.593.3.19	setUseClientMode . . . . .	2953
6.593.3.20	setWantClientAuth . . . . .	2953
6.593.3.21	shutdownInput . . . . .	2953
6.593.3.22	shutdownOutput . . . . .	2954
6.593.3.23	startHandshake . . . . .	2954
6.593.3.24	write . . . . .	2954
6.594	decaf::internal::net::ssl::openssl::OpenSSLSocketException Class Reference . . . . .	2955
6.594.1	Detailed Description . . . . .	2956
6.594.2	Constructor & Destructor Documentation . . . . .	2956
6.594.2.1	OpenSSLSocketException . . . . .	2956
6.594.2.2	OpenSSLSocketException . . . . .	2956
6.594.2.3	OpenSSLSocketException . . . . .	2956
6.594.2.4	OpenSSLSocketException . . . . .	2957
6.594.2.5	OpenSSLSocketException . . . . .	2957
6.594.2.6	OpenSSLSocketException . . . . .	2957
6.594.2.7	OpenSSLSocketException . . . . .	2958
6.594.2.8	~OpenSSLSocketException . . . . .	2958
6.594.3	Member Function Documentation . . . . .	2958
6.594.3.1	clone . . . . .	2958
6.594.3.2	getErrorString . . . . .	2958
6.595	decaf::internal::net::ssl::openssl::OpenSSLSocketFactory Class Reference . . . . .	2959
6.595.1	Detailed Description . . . . .	2962
6.595.2	Constructor & Destructor Documentation . . . . .	2962
6.595.2.1	OpenSSLSocketFactory . . . . .	2962
6.595.2.2	~OpenSSLSocketFactory . . . . .	2962
6.595.3	Member Function Documentation . . . . .	2962
6.595.3.1	createSocket . . . . .	2962
6.595.3.2	createSocket . . . . .	2962
6.595.3.3	createSocket . . . . .	2963
6.595.3.4	createSocket . . . . .	2964
6.595.3.5	createSocket . . . . .	2964
6.595.3.6	createSocket . . . . .	2965
6.595.3.7	getDefaultCipherSuites . . . . .	2965
6.595.3.8	getSupportedCipherSuites . . . . .	2966
6.596	decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream Class Reference . . . . .	2966
6.596.1	Detailed Description . . . . .	2967
6.596.2	Constructor & Destructor Documentation . . . . .	2967
6.596.2.1	OpenSSLSocketInputStream . . . . .	2967
6.596.2.2	~OpenSSLSocketInputStream . . . . .	2967
6.596.3	Member Function Documentation . . . . .	2967
6.596.3.1	available . . . . .	2967
6.596.3.2	close . . . . .	2968

6.596.3.3 doReadArrayBounded . . . . .	2968
6.596.3.4 doReadByte . . . . .	2968
6.596.3.5 skip . . . . .	2968
6.597decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream Class Reference . . . . .	2969
6.597.1 Detailed Description . . . . .	2970
6.597.2 Constructor & Destructor Documentation . . . . .	2970
6.597.2.1 OpenSSLSocketOutputStream . . . . .	2970
6.597.2.2 ~OpenSSLSocketOutputStream . . . . .	2970
6.597.3 Member Function Documentation . . . . .	2970
6.597.3.1 close . . . . .	2970
6.597.3.2 doWriteArrayBounded . . . . .	2971
6.597.3.3 doWriteByte . . . . .	2971
6.598activemq::wireformat::openwire::OpenWireFormat Class Reference . . . . .	2971
6.598.1 Constructor & Destructor Documentation . . . . .	2974
6.598.1.1 OpenWireFormat . . . . .	2974
6.598.1.2 ~OpenWireFormat . . . . .	2975
6.598.2 Member Function Documentation . . . . .	2975
6.598.2.1 addMarshaller . . . . .	2975
6.598.2.2 createNegotiator . . . . .	2975
6.598.2.3 destroyMarshalers . . . . .	2975
6.598.2.4 doUnmarshal . . . . .	2975
6.598.2.5 getCacheSize . . . . .	2976
6.598.2.6 getMaxInactivityDuration . . . . .	2976
6.598.2.7 getMaxInactivityDurationInitialDelay . . . . .	2976
6.598.2.8 getPreferredWireFormatInfo . . . . .	2976
6.598.2.9 getVersion . . . . .	2977
6.598.2.10hasNegotiator . . . . .	2977
6.598.2.11inReceive . . . . .	2977
6.598.2.12sCacheEnabled . . . . .	2977
6.598.2.13sSizePrefixDisabled . . . . .	2977
6.598.2.14sStackTraceEnabled . . . . .	2978
6.598.2.15sTcpNoDelayEnabled . . . . .	2978
6.598.2.16sTightEncodingEnabled . . . . .	2978
6.598.2.17looseMarshalNestedObject . . . . .	2978
6.598.2.18looseUnmarshalNestedObject . . . . .	2979
6.598.2.19marshal . . . . .	2979
6.598.2.20renegotiateWireFormat . . . . .	2980
6.598.2.21setCacheEnabled . . . . .	2980
6.598.2.22setCacheSize . . . . .	2980
6.598.2.23setMaxInactivityDuration . . . . .	2980
6.598.2.24setMaxInactivityDurationInitialDelay . . . . .	2981
6.598.2.25setPreferredWireFormatInfo . . . . .	2981
6.598.2.26setSizePrefixDisabled . . . . .	2981
6.598.2.27setStackTraceEnabled . . . . .	2981
6.598.2.28setTcpNoDelayEnabled . . . . .	2981
6.598.2.29setTightEncodingEnabled . . . . .	2982
6.598.2.30setVersion . . . . .	2982
6.598.2.31tightMarshalNestedObject1 . . . . .	2982
6.598.2.32tightMarshalNestedObject2 . . . . .	2983

6.598.2.33	tightUnmarshalNestedObject . . . . .	2983
6.598.2.34	unmarshal . . . . .	2983
6.598.3	Field Documentation . . . . .	2984
6.598.3.1	DEFAULT_VERSION . . . . .	2984
6.598.3.2	NULL_TYPE . . . . .	2984
6.599	activemq::wireformat::openwire::OpenWireFormatFactory Class Reference . . . . .	2984
6.599.1	Constructor & Destructor Documentation . . . . .	2985
6.599.1.1	OpenWireFormatFactory . . . . .	2985
6.599.1.2	~OpenWireFormatFactory . . . . .	2985
6.599.2	Member Function Documentation . . . . .	2985
6.599.2.1	createWireFormat . . . . .	2985
6.600	activemq::wireformat::openwire::OpenWireFormatNegotiator Class Reference . . . . .	2985
6.600.1	Constructor & Destructor Documentation . . . . .	2986
6.600.1.1	OpenWireFormatNegotiator . . . . .	2986
6.600.1.2	~OpenWireFormatNegotiator . . . . .	2987
6.600.2	Member Function Documentation . . . . .	2987
6.600.2.1	close . . . . .	2987
6.600.2.2	onCommand . . . . .	2987
6.600.2.3	oneway . . . . .	2987
6.600.2.4	onTransportException . . . . .	2988
6.600.2.5	request . . . . .	2988
6.600.2.6	request . . . . .	2989
6.600.2.7	start . . . . .	2989
6.601	activemq::wireformat::openwire::OpenWireResponseBuilder Class Reference . . . . .	2989
6.601.1	Constructor & Destructor Documentation . . . . .	2990
6.601.1.1	OpenWireResponseBuilder . . . . .	2990
6.601.1.2	~OpenWireResponseBuilder . . . . .	2990
6.601.2	Member Function Documentation . . . . .	2990
6.601.2.1	buildIncomingCommands . . . . .	2990
6.601.2.2	buildResponse . . . . .	2991
6.602	decaf::io::OutputStream Class Reference . . . . .	2991
6.602.1	Detailed Description . . . . .	2993
6.602.2	Constructor & Destructor Documentation . . . . .	2993
6.602.2.1	OutputStream . . . . .	2993
6.602.2.2	~OutputStream . . . . .	2993
6.602.3	Member Function Documentation . . . . .	2993
6.602.3.1	close . . . . .	2993
6.602.3.2	doWriteArray . . . . .	2994
6.602.3.3	doWriteArrayBounded . . . . .	2994
6.602.3.4	doWriteByte . . . . .	2994
6.602.3.5	flush . . . . .	2994
6.602.3.6	lock . . . . .	2995
6.602.3.7	notify . . . . .	2995
6.602.3.8	notifyAll . . . . .	2995
6.602.3.9	toString . . . . .	2995
6.602.3.10	tryLock . . . . .	2996
6.602.3.11	unlock . . . . .	2996

6.602.3.12wait . . . . .	2996
6.602.3.13wait . . . . .	2997
6.602.3.14wait . . . . .	2997
6.602.3.15write . . . . .	2998
6.602.3.16write . . . . .	2998
6.602.3.17write . . . . .	2999
6.603decaf::io::OutputStreamWriter Class Reference . . . . .	2999
6.603.1 Detailed Description . . . . .	3000
6.603.2 Constructor & Destructor Documentation . . . . .	3000
6.603.2.1 OutputStreamWriter . . . . .	3000
6.603.2.2 ~OutputStreamWriter . . . . .	3001
6.603.3 Member Function Documentation . . . . .	3001
6.603.3.1 checkClosed . . . . .	3001
6.603.3.2 close . . . . .	3001
6.603.3.3 doWriteArrayBounded . . . . .	3001
6.603.3.4 flush . . . . .	3001
6.604activemq::commands::PartialCommand Class Reference . . . . .	3002
6.604.1 Constructor & Destructor Documentation . . . . .	3003
6.604.1.1 PartialCommand . . . . .	3003
6.604.1.2 ~PartialCommand . . . . .	3003
6.604.2 Member Function Documentation . . . . .	3003
6.604.2.1 cloneDataStructure . . . . .	3003
6.604.2.2 copyDataStructure . . . . .	3003
6.604.2.3 equals . . . . .	3004
6.604.2.4 getCommandId . . . . .	3004
6.604.2.5 getData . . . . .	3004
6.604.2.6 getData . . . . .	3004
6.604.2.7 getDataStructureType . . . . .	3004
6.604.2.8 setCommandId . . . . .	3005
6.604.2.9 setData . . . . .	3005
6.604.2.10toString . . . . .	3005
6.604.3 Field Documentation . . . . .	3005
6.604.3.1 commandId . . . . .	3005
6.604.3.2 data . . . . .	3005
6.604.3.3 ID_PARTIALCOMMAND . . . . .	3005
6.605activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller	
Class Reference . . . . .	3005
6.605.1 Detailed Description . . . . .	3006
6.605.2 Constructor & Destructor Documentation . . . . .	3007
6.605.2.1 PartialCommandMarshaller . . . . .	3007
6.605.2.2 ~PartialCommandMarshaller . . . . .	3007
6.605.3 Member Function Documentation . . . . .	3007
6.605.3.1 createObject . . . . .	3007
6.605.3.2 getDataStructureType . . . . .	3007
6.605.3.3 looseMarshal . . . . .	3007
6.605.3.4 looseUnmarshal . . . . .	3008
6.605.3.5 tightMarshal1 . . . . .	3008
6.605.3.6 tightMarshal2 . . . . .	3009
6.605.3.7 tightUnmarshal . . . . .	3009

6.606activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller	
Class Reference . . . . .	3010
6.606.1 Detailed Description . . . . .	3011
6.606.2 Constructor & Destructor Documentation . . . . .	3011
6.606.2.1 PartialCommandMarshaller . . . . .	3011
6.606.2.2 ~PartialCommandMarshaller . . . . .	3011
6.606.3 Member Function Documentation . . . . .	3011
6.606.3.1 createObject . . . . .	3011
6.606.3.2 getDataStructureType . . . . .	3011
6.606.3.3 looseMarshal . . . . .	3012
6.606.3.4 looseUnmarshal . . . . .	3012
6.606.3.5 tightMarshal1 . . . . .	3013
6.606.3.6 tightMarshal2 . . . . .	3013
6.606.3.7 tightUnmarshal . . . . .	3014
6.607activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller	
Class Reference . . . . .	3014
6.607.1 Detailed Description . . . . .	3015
6.607.2 Constructor & Destructor Documentation . . . . .	3016
6.607.2.1 PartialCommandMarshaller . . . . .	3016
6.607.2.2 ~PartialCommandMarshaller . . . . .	3016
6.607.3 Member Function Documentation . . . . .	3016
6.607.3.1 createObject . . . . .	3016
6.607.3.2 getDataStructureType . . . . .	3016
6.607.3.3 looseMarshal . . . . .	3016
6.607.3.4 looseUnmarshal . . . . .	3017
6.607.3.5 tightMarshal1 . . . . .	3017
6.607.3.6 tightMarshal2 . . . . .	3018
6.607.3.7 tightUnmarshal . . . . .	3018
6.608activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller	
Class Reference . . . . .	3019
6.608.1 Detailed Description . . . . .	3020
6.608.2 Constructor & Destructor Documentation . . . . .	3020
6.608.2.1 PartialCommandMarshaller . . . . .	3020
6.608.2.2 ~PartialCommandMarshaller . . . . .	3020
6.608.3 Member Function Documentation . . . . .	3020
6.608.3.1 createObject . . . . .	3020
6.608.3.2 getDataStructureType . . . . .	3020
6.608.3.3 looseMarshal . . . . .	3021
6.608.3.4 looseUnmarshal . . . . .	3021
6.608.3.5 tightMarshal1 . . . . .	3022
6.608.3.6 tightMarshal2 . . . . .	3022
6.608.3.7 tightUnmarshal . . . . .	3023
6.609activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller	
Class Reference . . . . .	3023
6.609.1 Detailed Description . . . . .	3024
6.609.2 Constructor & Destructor Documentation . . . . .	3025
6.609.2.1 PartialCommandMarshaller . . . . .	3025
6.609.2.2 ~PartialCommandMarshaller . . . . .	3025
6.609.3 Member Function Documentation . . . . .	3025
6.609.3.1 createObject . . . . .	3025

6.609.3.2	getDataStructureType	3025
6.609.3.3	looseMarshal	3025
6.609.3.4	looseUnmarshal	3026
6.609.3.5	tightMarshal1	3026
6.609.3.6	tightMarshal2	3027
6.609.3.7	tightUnmarshal	3027
6.610	activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	
	Class Reference	3028
6.610.1	Detailed Description	3029
6.610.2	Constructor & Destructor Documentation	3029
6.610.2.1	PartialCommandMarshaller	3029
6.610.2.2	~PartialCommandMarshaller	3029
6.610.3	Member Function Documentation	3029
6.610.3.1	createObject	3029
6.610.3.2	getDataStructureType	3029
6.610.3.3	looseMarshal	3030
6.610.3.4	looseUnmarshal	3030
6.610.3.5	tightMarshal1	3031
6.610.3.6	tightMarshal2	3031
6.610.3.7	tightUnmarshal	3032
6.611	decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference	3032
6.611.1	Detailed Description	3034
6.611.2	Member Typedef Documentation	3035
6.611.2.1	CounterType	3035
6.611.2.2	PointerType	3035
6.611.2.3	ReferenceType	3035
6.611.3	Constructor & Destructor Documentation	3035
6.611.3.1	Pointer	3035
6.611.3.2	Pointer	3035
6.611.3.3	Pointer	3035
6.611.3.4	Pointer	3036
6.611.3.5	Pointer	3036
6.611.3.6	Pointer	3036
6.611.3.7	~Pointer	3037
6.611.4	Member Function Documentation	3037
6.611.4.1	dynamicCast	3037
6.611.4.2	get	3037
6.611.4.3	operator!	3037
6.611.4.4	operator!=	3037
6.611.4.5	operator*	3037
6.611.4.6	operator*	3037
6.611.4.7	operator->	3038
6.611.4.8	operator->	3038
6.611.4.9	operator=	3038
6.611.4.10	operator=	3039
6.611.4.11	operator==	3039
6.611.4.12	release	3039
6.611.4.13	reset	3039
6.611.4.14	staticCast	3040
6.611.4.15	swap	3040

6.611.5 Friends And Related Function Documentation . . . . .	3040
6.611.5.1 operator!= . . . . .	3040
6.611.5.2 operator!= . . . . .	3040
6.611.5.3 operator== . . . . .	3040
6.611.5.4 operator== . . . . .	3040
6.612decaf::lang::PointerComparator< T, R > Class Template Reference . . . . .	3040
6.612.1 Detailed Description . . . . .	3041
6.612.2 Member Function Documentation . . . . .	3041
6.612.2.1 compare . . . . .	3041
6.612.2.2 operator() . . . . .	3041
6.613activemq::cmsutil::PooledSession Class Reference . . . . .	3041
6.613.1 Detailed Description . . . . .	3044
6.613.2 Constructor & Destructor Documentation . . . . .	3045
6.613.2.1 PooledSession . . . . .	3045
6.613.2.2 PooledSession . . . . .	3045
6.613.2.3 ~PooledSession . . . . .	3045
6.613.3 Member Function Documentation . . . . .	3045
6.613.3.1 close . . . . .	3045
6.613.3.2 commit . . . . .	3045
6.613.3.3 createBrowser . . . . .	3045
6.613.3.4 createBrowser . . . . .	3046
6.613.3.5 createBytesMessage . . . . .	3046
6.613.3.6 createBytesMessage . . . . .	3047
6.613.3.7 createCachedConsumer . . . . .	3047
6.613.3.8 createCachedProducer . . . . .	3047
6.613.3.9 createConsumer . . . . .	3048
6.613.3.10createConsumer . . . . .	3048
6.613.3.11createConsumer . . . . .	3049
6.613.3.12createDurableConsumer . . . . .	3049
6.613.3.13createMapMessage . . . . .	3050
6.613.3.14createMessage . . . . .	3050
6.613.3.15createProducer . . . . .	3051
6.613.3.16createQueue . . . . .	3051
6.613.3.17createStreamMessage . . . . .	3052
6.613.3.18createTemporaryQueue . . . . .	3052
6.613.3.19createTemporaryTopic . . . . .	3052
6.613.3.20createTextMessage . . . . .	3052
6.613.3.21createTextMessage . . . . .	3053
6.613.3.22createTopic . . . . .	3053
6.613.3.23getAcknowledgeMode . . . . .	3053
6.613.3.24getSession . . . . .	3054
6.613.3.25getSession . . . . .	3054
6.613.3.26isTransacted . . . . .	3054
6.613.3.27operator= . . . . .	3054
6.613.3.28recover . . . . .	3054
6.613.3.29rollback . . . . .	3055
6.613.3.30unsubscribe . . . . .	3055
6.614decaf::util::concurrent::PooledThread Class Reference . . . . .	3056
6.614.1 Constructor & Destructor Documentation . . . . .	3057
6.614.1.1 PooledThread . . . . .	3057

6.614.1.2 ~PooledThread . . . . .	3057
6.614.2 Member Function Documentation . . . . .	3057
6.614.2.1 getPooledThreadListener . . . . .	3057
6.614.2.2 isBusy . . . . .	3057
6.614.2.3 run . . . . .	3057
6.614.2.4 setPooledThreadListener . . . . .	3057
6.614.2.5 stop . . . . .	3058
6.615decaf::util::concurrent::PooledThreadListener Class Reference . . . . .	3058
6.615.1 Detailed Description . . . . .	3059
6.615.2 Constructor & Destructor Documentation . . . . .	3059
6.615.2.1 ~PooledThreadListener . . . . .	3059
6.615.3 Member Function Documentation . . . . .	3059
6.615.3.1 onTaskCompleted . . . . .	3059
6.615.3.2 onTaskException . . . . .	3059
6.615.3.3 onTaskStarted . . . . .	3059
6.616decaf::net::PortUnreachableException Class Reference . . . . .	3060
6.616.1 Constructor & Destructor Documentation . . . . .	3061
6.616.1.1 PortUnreachableException . . . . .	3061
6.616.1.2 PortUnreachableException . . . . .	3061
6.616.1.3 PortUnreachableException . . . . .	3061
6.616.1.4 PortUnreachableException . . . . .	3061
6.616.1.5 PortUnreachableException . . . . .	3061
6.616.1.6 PortUnreachableException . . . . .	3062
6.616.1.7 ~PortUnreachableException . . . . .	3062
6.616.2 Member Function Documentation . . . . .	3062
6.616.2.1 clone . . . . .	3062
6.617activemq::core::PrefetchPolicy Class Reference . . . . .	3062
6.617.1 Detailed Description . . . . .	3064
6.617.2 Constructor & Destructor Documentation . . . . .	3064
6.617.2.1 PrefetchPolicy . . . . .	3064
6.617.2.2 ~PrefetchPolicy . . . . .	3064
6.617.3 Member Function Documentation . . . . .	3064
6.617.3.1 clone . . . . .	3064
6.617.3.2 configure . . . . .	3064
6.617.3.3 getDurableTopicPrefetch . . . . .	3065
6.617.3.4 getMaxPrefetchLimit . . . . .	3065
6.617.3.5 getQueueBrowserPrefetch . . . . .	3065
6.617.3.6 getQueuePrefetch . . . . .	3065
6.617.3.7 getTopicPrefetch . . . . .	3066
6.617.3.8 setDurableTopicPrefetch . . . . .	3066
6.617.3.9 setQueueBrowserPrefetch . . . . .	3066
6.617.3.10setQueuePrefetch . . . . .	3066
6.617.3.11setTopicPrefetch . . . . .	3067
6.618activemq::util::PrimitiveList Class Reference . . . . .	3067
6.618.1 Detailed Description . . . . .	3070
6.618.2 Constructor & Destructor Documentation . . . . .	3070
6.618.2.1 PrimitiveList . . . . .	3070
6.618.2.2 ~PrimitiveList . . . . .	3070
6.618.2.3 PrimitiveList . . . . .	3070
6.618.2.4 PrimitiveList . . . . .	3070



6.618.3 Member Function Documentation . . . . .	3070
6.618.3.1 getBool . . . . .	3070
6.618.3.2 getByte . . . . .	3071
6.618.3.3 getByteArray . . . . .	3071
6.618.3.4 getChar . . . . .	3072
6.618.3.5 getDouble . . . . .	3072
6.618.3.6 getFloat . . . . .	3073
6.618.3.7 getInt . . . . .	3073
6.618.3.8 getLong . . . . .	3074
6.618.3.9 getShort . . . . .	3074
6.618.3.10 getString . . . . .	3075
6.618.3.11 setBool . . . . .	3075
6.618.3.12 setByte . . . . .	3075
6.618.3.13 setByteArray . . . . .	3076
6.618.3.14 setChar . . . . .	3076
6.618.3.15 setDouble . . . . .	3077
6.618.3.16 setFloat . . . . .	3077
6.618.3.17 setInt . . . . .	3077
6.618.3.18 setLong . . . . .	3078
6.618.3.19 setShort . . . . .	3078
6.618.3.20 setString . . . . .	3078
6.618.3.21 toString . . . . .	3079
6.619 activemq::util::PrimitiveMap Class Reference . . . . .	3079
6.619.1 Detailed Description . . . . .	3081
6.619.2 Constructor & Destructor Documentation . . . . .	3082
6.619.2.1 PrimitiveMap . . . . .	3082
6.619.2.2 ~PrimitiveMap . . . . .	3082
6.619.2.3 PrimitiveMap . . . . .	3082
6.619.2.4 PrimitiveMap . . . . .	3082
6.619.3 Member Function Documentation . . . . .	3082
6.619.3.1 getBool . . . . .	3082
6.619.3.2 getByte . . . . .	3083
6.619.3.3 getByteArray . . . . .	3083
6.619.3.4 getChar . . . . .	3084
6.619.3.5 getDouble . . . . .	3084
6.619.3.6 getFloat . . . . .	3085
6.619.3.7 getInt . . . . .	3085
6.619.3.8 getLong . . . . .	3086
6.619.3.9 getShort . . . . .	3086
6.619.3.10 getString . . . . .	3087
6.619.3.11 setBool . . . . .	3087
6.619.3.12 setByte . . . . .	3087
6.619.3.13 setByteArray . . . . .	3088
6.619.3.14 setChar . . . . .	3088
6.619.3.15 setDouble . . . . .	3088
6.619.3.16 setFloat . . . . .	3088
6.619.3.17 setInt . . . . .	3089
6.619.3.18 setLong . . . . .	3089
6.619.3.19 setShort . . . . .	3089
6.619.3.20 setString . . . . .	3089

6.619.3.2	toString	3090
6.620	activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller Class Reference	3090
6.620.1	Detailed Description	3092
6.620.2	Constructor & Destructor Documentation	3092
6.620.2.1	PrimitiveTypesMarshaller	3092
6.620.2.2	~PrimitiveTypesMarshaller	3092
6.620.3	Member Function Documentation	3092
6.620.3.1	marshal	3092
6.620.3.2	marshal	3092
6.620.3.3	marshalList	3093
6.620.3.4	marshalMap	3093
6.620.3.5	marshalPrimitive	3093
6.620.3.6	marshalPrimitiveList	3094
6.620.3.7	marshalPrimitiveMap	3094
6.620.3.8	unmarshal	3094
6.620.3.9	unmarshal	3095
6.620.3.10	unmarshalList	3095
6.620.3.11	unmarshalMap	3095
6.620.3.12	unmarshalPrimitive	3096
6.620.3.13	unmarshalPrimitiveList	3096
6.620.3.14	unmarshalPrimitiveMap	3096
6.621	activemq::util::PrimitiveValueNode::PrimitiveValue Union Reference	3097
6.621.1	Detailed Description	3097
6.621.2	Field Documentation	3098
6.621.2.1	boolValue	3098
6.621.2.2	byteArrayValue	3098
6.621.2.3	byteValue	3098
6.621.2.4	charValue	3098
6.621.2.5	doubleValue	3098
6.621.2.6	floatValue	3098
6.621.2.7	intValue	3098
6.621.2.8	listValue	3098
6.621.2.9	longValue	3098
6.621.2.10	mapValue	3098
6.621.2.11	shortValue	3098
6.621.2.12	stringValue	3098
6.622	activemq::util::PrimitiveValueConverter Class Reference	3098
6.622.1	Detailed Description	3099
6.622.2	Constructor & Destructor Documentation	3099
6.622.2.1	PrimitiveValueConverter	3099
6.622.2.2	~PrimitiveValueConverter	3099
6.622.3	Member Function Documentation	3099
6.622.3.1	convert	3099
6.623	activemq::util::PrimitiveValueNode Class Reference	3099
6.623.1	Detailed Description	3103
6.623.2	Member Enumeration Documentation	3104
6.623.2.1	PrimitiveType	3104
6.623.3	Constructor & Destructor Documentation	3104
6.623.3.1	PrimitiveValueNode	3104

6.623.3.2 PrimitiveValueNode . . . . .	3104
6.623.3.3 PrimitiveValueNode . . . . .	3105
6.623.3.4 PrimitiveValueNode . . . . .	3105
6.623.3.5 PrimitiveValueNode . . . . .	3105
6.623.3.6 PrimitiveValueNode . . . . .	3105
6.623.3.7 PrimitiveValueNode . . . . .	3105
6.623.3.8 PrimitiveValueNode . . . . .	3105
6.623.3.9 PrimitiveValueNode . . . . .	3106
6.623.3.10 PrimitiveValueNode . . . . .	3106
6.623.3.11 PrimitiveValueNode . . . . .	3106
6.623.3.12 PrimitiveValueNode . . . . .	3106
6.623.3.13 PrimitiveValueNode . . . . .	3106
6.623.3.14 PrimitiveValueNode . . . . .	3107
6.623.3.15 PrimitiveValueNode . . . . .	3107
6.623.3.16 ~PrimitiveValueNode . . . . .	3107
6.623.4 Member Function Documentation . . . . .	3107
6.623.4.1 clear . . . . .	3107
6.623.4.2 getBool . . . . .	3107
6.623.4.3 getByte . . . . .	3107
6.623.4.4 getByteArray . . . . .	3108
6.623.4.5 getChar . . . . .	3108
6.623.4.6 getDouble . . . . .	3108
6.623.4.7 getFloat . . . . .	3109
6.623.4.8 getInt . . . . .	3109
6.623.4.9 getList . . . . .	3109
6.623.4.10 getLong . . . . .	3110
6.623.4.11 getMap . . . . .	3110
6.623.4.12 getShort . . . . .	3110
6.623.4.13 getString . . . . .	3110
6.623.4.14 getType . . . . .	3111
6.623.4.15 getValue . . . . .	3111
6.623.4.16 operator= . . . . .	3111
6.623.4.17 operator== . . . . .	3111
6.623.4.18 setBool . . . . .	3112
6.623.4.19 setByte . . . . .	3112
6.623.4.20 setByteArray . . . . .	3112
6.623.4.21 setChar . . . . .	3112
6.623.4.22 setDouble . . . . .	3112
6.623.4.23 setFloat . . . . .	3113
6.623.4.24 setInt . . . . .	3113
6.623.4.25 setList . . . . .	3113
6.623.4.26 setLong . . . . .	3113
6.623.4.27 setMap . . . . .	3113
6.623.4.28 setShort . . . . .	3114
6.623.4.29 setString . . . . .	3114
6.623.4.30 setValue . . . . .	3114
6.623.4.31 toString . . . . .	3114
6.624 decaf::security::Principal Class Reference . . . . .	3114
6.624.1 Detailed Description . . . . .	3115
6.624.2 Constructor & Destructor Documentation . . . . .	3115

6.624.2.1 ~Principal . . . . .	3115
6.624.3 Member Function Documentation . . . . .	3115
6.624.3.1 equals . . . . .	3115
6.624.3.2 getName . . . . .	3115
6.625decaf::util::PriorityQueue< E > Class Template Reference . . . . .	3116
6.625.1 Detailed Description . . . . .	3118
6.625.2 Constructor & Destructor Documentation . . . . .	3118
6.625.2.1 PriorityQueue . . . . .	3118
6.625.2.2 PriorityQueue . . . . .	3118
6.625.2.3 PriorityQueue . . . . .	3119
6.625.2.4 PriorityQueue . . . . .	3119
6.625.2.5 PriorityQueue . . . . .	3119
6.625.2.6 ~PriorityQueue . . . . .	3120
6.625.3 Member Function Documentation . . . . .	3120
6.625.3.1 add . . . . .	3120
6.625.3.2 clear . . . . .	3120
6.625.3.3 comparator . . . . .	3120
6.625.3.4 iterator . . . . .	3121
6.625.3.5 iterator . . . . .	3121
6.625.3.6 offer . . . . .	3121
6.625.3.7 operator= . . . . .	3122
6.625.3.8 operator= . . . . .	3122
6.625.3.9 peek . . . . .	3122
6.625.3.10poll . . . . .	3122
6.625.3.11remove . . . . .	3123
6.625.3.12remove . . . . .	3123
6.625.3.13size . . . . .	3124
6.625.4 Friends And Related Function Documentation . . . . .	3124
6.625.4.1 PriorityQueueIterator . . . . .	3124
6.626activemq::commands::ProducerAck Class Reference . . . . .	3124
6.626.1 Constructor & Destructor Documentation . . . . .	3126
6.626.1.1 ProducerAck . . . . .	3126
6.626.1.2 ~ProducerAck . . . . .	3126
6.626.2 Member Function Documentation . . . . .	3126
6.626.2.1 cloneDataStructure . . . . .	3126
6.626.2.2 copyDataStructure . . . . .	3126
6.626.2.3 equals . . . . .	3126
6.626.2.4 getDataStructureType . . . . .	3127
6.626.2.5 getProducerId . . . . .	3127
6.626.2.6 getProducerId . . . . .	3127
6.626.2.7 getSize . . . . .	3127
6.626.2.8 isProducerAck . . . . .	3127
6.626.2.9 setProducerId . . . . .	3127
6.626.2.10setSize . . . . .	3127
6.626.2.11toString . . . . .	3127
6.626.2.12visit . . . . .	3128
6.626.3 Field Documentation . . . . .	3128
6.626.3.1 ID_PRODUCERACK . . . . .	3128
6.626.3.2 producerId . . . . .	3128
6.626.3.3 size . . . . .	3128

6.627activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller	
Class Reference . . . . .	3128
6.627.1 Detailed Description . . . . .	3129
6.627.2 Constructor & Destructor Documentation . . . . .	3130
6.627.2.1 ProducerAckMarshaller . . . . .	3130
6.627.2.2 ~ProducerAckMarshaller . . . . .	3130
6.627.3 Member Function Documentation . . . . .	3130
6.627.3.1 createObject . . . . .	3130
6.627.3.2 getDataStructureType . . . . .	3130
6.627.3.3 looseMarshal . . . . .	3130
6.627.3.4 looseUnmarshal . . . . .	3131
6.627.3.5 tightMarshal1 . . . . .	3131
6.627.3.6 tightMarshal2 . . . . .	3132
6.627.3.7 tightUnmarshal . . . . .	3132
6.628activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller	
Class Reference . . . . .	3133
6.628.1 Detailed Description . . . . .	3134
6.628.2 Constructor & Destructor Documentation . . . . .	3134
6.628.2.1 ProducerAckMarshaller . . . . .	3134
6.628.2.2 ~ProducerAckMarshaller . . . . .	3134
6.628.3 Member Function Documentation . . . . .	3134
6.628.3.1 createObject . . . . .	3134
6.628.3.2 getDataStructureType . . . . .	3134
6.628.3.3 looseMarshal . . . . .	3134
6.628.3.4 looseUnmarshal . . . . .	3135
6.628.3.5 tightMarshal1 . . . . .	3135
6.628.3.6 tightMarshal2 . . . . .	3136
6.628.3.7 tightUnmarshal . . . . .	3136
6.629activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller	
Class Reference . . . . .	3137
6.629.1 Detailed Description . . . . .	3138
6.629.2 Constructor & Destructor Documentation . . . . .	3138
6.629.2.1 ProducerAckMarshaller . . . . .	3138
6.629.2.2 ~ProducerAckMarshaller . . . . .	3138
6.629.3 Member Function Documentation . . . . .	3138
6.629.3.1 createObject . . . . .	3138
6.629.3.2 getDataStructureType . . . . .	3138
6.629.3.3 looseMarshal . . . . .	3139
6.629.3.4 looseUnmarshal . . . . .	3139
6.629.3.5 tightMarshal1 . . . . .	3140
6.629.3.6 tightMarshal2 . . . . .	3140
6.629.3.7 tightUnmarshal . . . . .	3141
6.630activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller	
Class Reference . . . . .	3141
6.630.1 Detailed Description . . . . .	3142
6.630.2 Constructor & Destructor Documentation . . . . .	3142
6.630.2.1 ProducerAckMarshaller . . . . .	3142
6.630.2.2 ~ProducerAckMarshaller . . . . .	3142
6.630.3 Member Function Documentation . . . . .	3142
6.630.3.1 createObject . . . . .	3142

6.630.3.2	getDataStructureType	3143
6.630.3.3	looseMarshal	3143
6.630.3.4	looseUnmarshal	3143
6.630.3.5	tightMarshal1	3144
6.630.3.6	tightMarshal2	3144
6.630.3.7	tightUnmarshal	3145
6.631	activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller	
	Class Reference	3145
6.631.1	Detailed Description	3146
6.631.2	Constructor & Destructor Documentation	3147
6.631.2.1	ProducerAckMarshaller	3147
6.631.2.2	~ProducerAckMarshaller	3147
6.631.3	Member Function Documentation	3147
6.631.3.1	createObject	3147
6.631.3.2	getDataStructureType	3147
6.631.3.3	looseMarshal	3147
6.631.3.4	looseUnmarshal	3148
6.631.3.5	tightMarshal1	3148
6.631.3.6	tightMarshal2	3149
6.631.3.7	tightUnmarshal	3149
6.632	activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller	
	Class Reference	3150
6.632.1	Detailed Description	3151
6.632.2	Constructor & Destructor Documentation	3151
6.632.2.1	ProducerAckMarshaller	3151
6.632.2.2	~ProducerAckMarshaller	3151
6.632.3	Member Function Documentation	3151
6.632.3.1	createObject	3151
6.632.3.2	getDataStructureType	3151
6.632.3.3	looseMarshal	3151
6.632.3.4	looseUnmarshal	3152
6.632.3.5	tightMarshal1	3152
6.632.3.6	tightMarshal2	3153
6.632.3.7	tightUnmarshal	3153
6.633	activemq::cmsutil::ProducerCallback Class Reference	3154
6.633.1	Detailed Description	3154
6.633.2	Constructor & Destructor Documentation	3154
6.633.2.1	~ProducerCallback	3154
6.633.3	Member Function Documentation	3154
6.633.3.1	doInCms	3154
6.634	activemq::cmsutil::CmsTemplate::ProducerExecutor Class Reference	3155
6.634.1	Constructor & Destructor Documentation	3156
6.634.1.1	ProducerExecutor	3156
6.634.1.2	ProducerExecutor	3156
6.634.1.3	~ProducerExecutor	3156
6.634.2	Member Function Documentation	3156
6.634.2.1	doInCms	3156
6.634.2.2	getDestination	3156
6.634.2.3	operator=	3156
6.634.3	Field Documentation	3156

6.634.3.1 action . . . . .	3156
6.634.3.2 destination . . . . .	3156
6.634.3.3 parent . . . . .	3156
6.635activemq::commands::ProducerId Class Reference . . . . .	3157
6.635.1 Member Typedef Documentation . . . . .	3158
6.635.1.1 COMPARATOR . . . . .	3158
6.635.2 Constructor & Destructor Documentation . . . . .	3158
6.635.2.1 ProducerId . . . . .	3158
6.635.2.2 ProducerId . . . . .	3158
6.635.2.3 ProducerId . . . . .	3158
6.635.2.4 ProducerId . . . . .	3158
6.635.2.5 ~ProducerId . . . . .	3158
6.635.3 Member Function Documentation . . . . .	3158
6.635.3.1 cloneDataStructure . . . . .	3158
6.635.3.2 compareTo . . . . .	3159
6.635.3.3 copyDataStructure . . . . .	3159
6.635.3.4 equals . . . . .	3159
6.635.3.5 equals . . . . .	3159
6.635.3.6 getConnectionId . . . . .	3159
6.635.3.7 getConnectionId . . . . .	3159
6.635.3.8 getDataStructureType . . . . .	3159
6.635.3.9 getParentId . . . . .	3160
6.635.3.10getSessionId . . . . .	3160
6.635.3.11getValue . . . . .	3160
6.635.3.12operator< . . . . .	3160
6.635.3.13operator= . . . . .	3160
6.635.3.14operator== . . . . .	3160
6.635.3.15setConnectionId . . . . .	3160
6.635.3.16setProducerSessionKey . . . . .	3160
6.635.3.17setSessionId . . . . .	3160
6.635.3.18setValue . . . . .	3160
6.635.3.19toString . . . . .	3160
6.635.4 Field Documentation . . . . .	3161
6.635.4.1 connectionId . . . . .	3161
6.635.4.2 ID_PRODUCERID . . . . .	3161
6.635.4.3 sessionId . . . . .	3161
6.635.4.4 value . . . . .	3161
6.636activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller Class Reference . . . . .	3161
6.636.1 Detailed Description . . . . .	3162
6.636.2 Constructor & Destructor Documentation . . . . .	3162
6.636.2.1 ProducerIdMarshaller . . . . .	3162
6.636.2.2 ~ProducerIdMarshaller . . . . .	3162
6.636.3 Member Function Documentation . . . . .	3162
6.636.3.1 createObject . . . . .	3162
6.636.3.2 getDataStructureType . . . . .	3163
6.636.3.3 looseMarshal . . . . .	3163
6.636.3.4 looseUnmarshal . . . . .	3163
6.636.3.5 tightMarshal1 . . . . .	3164
6.636.3.6 tightMarshal2 . . . . .	3164

6.636.3.7	tightUnmarshal	3165
6.637	activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller Class	
	Reference	3165
6.637.1	Detailed Description	3166
6.637.2	Constructor & Destructor Documentation	3166
6.637.2.1	ProducerIdMarshaller	3166
6.637.2.2	~ProducerIdMarshaller	3166
6.637.3	Member Function Documentation	3166
6.637.3.1	createObject	3166
6.637.3.2	getDataStructureType	3167
6.637.3.3	looseMarshal	3167
6.637.3.4	looseUnmarshal	3167
6.637.3.5	tightMarshal1	3168
6.637.3.6	tightMarshal2	3168
6.637.3.7	tightUnmarshal	3169
6.638	activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller Class	
	Reference	3169
6.638.1	Detailed Description	3170
6.638.2	Constructor & Destructor Documentation	3170
6.638.2.1	ProducerIdMarshaller	3170
6.638.2.2	~ProducerIdMarshaller	3170
6.638.3	Member Function Documentation	3170
6.638.3.1	createObject	3170
6.638.3.2	getDataStructureType	3171
6.638.3.3	looseMarshal	3171
6.638.3.4	looseUnmarshal	3171
6.638.3.5	tightMarshal1	3172
6.638.3.6	tightMarshal2	3172
6.638.3.7	tightUnmarshal	3173
6.639	activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller Class	
	Reference	3173
6.639.1	Detailed Description	3174
6.639.2	Constructor & Destructor Documentation	3174
6.639.2.1	ProducerIdMarshaller	3174
6.639.2.2	~ProducerIdMarshaller	3174
6.639.3	Member Function Documentation	3174
6.639.3.1	createObject	3174
6.639.3.2	getDataStructureType	3175
6.639.3.3	looseMarshal	3175
6.639.3.4	looseUnmarshal	3175
6.639.3.5	tightMarshal1	3176
6.639.3.6	tightMarshal2	3176
6.639.3.7	tightUnmarshal	3177
6.640	activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller Class	
	Reference	3177
6.640.1	Detailed Description	3178
6.640.2	Constructor & Destructor Documentation	3178
6.640.2.1	ProducerIdMarshaller	3178
6.640.2.2	~ProducerIdMarshaller	3178
6.640.3	Member Function Documentation	3178



6.640.3.1	createObject	3178
6.640.3.2	getDataStructureType	3179
6.640.3.3	looseMarshal	3179
6.640.3.4	looseUnmarshal	3179
6.640.3.5	tightMarshal1	3180
6.640.3.6	tightMarshal2	3180
6.640.3.7	tightUnmarshal	3181
6.641	activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller Class	
Reference		3181
6.641.1	Detailed Description	3182
6.641.2	Constructor & Destructor Documentation	3182
6.641.2.1	ProducerIdMarshaller	3182
6.641.2.2	~ProducerIdMarshaller	3182
6.641.3	Member Function Documentation	3182
6.641.3.1	createObject	3182
6.641.3.2	getDataStructureType	3183
6.641.3.3	looseMarshal	3183
6.641.3.4	looseUnmarshal	3183
6.641.3.5	tightMarshal1	3184
6.641.3.6	tightMarshal2	3184
6.641.3.7	tightUnmarshal	3185
6.642	activemq::commands::ProducerInfo Class Reference	3185
6.642.1	Constructor & Destructor Documentation	3187
6.642.1.1	ProducerInfo	3187
6.642.1.2	~ProducerInfo	3187
6.642.2	Member Function Documentation	3187
6.642.2.1	cloneDataStructure	3187
6.642.2.2	copyDataStructure	3187
6.642.2.3	createRemoveCommand	3187
6.642.2.4	equals	3187
6.642.2.5	getBrokerPath	3188
6.642.2.6	getBrokerPath	3188
6.642.2.7	getDataStructureType	3188
6.642.2.8	getDestination	3188
6.642.2.9	getDestination	3188
6.642.2.10	getProducerId	3188
6.642.2.11	getProducerId	3188
6.642.2.12	getWindowSize	3188
6.642.2.13	isDispatchAsync	3188
6.642.2.14	isProducerInfo	3188
6.642.2.15	setBrokerPath	3189
6.642.2.16	setDestination	3189
6.642.2.17	setDispatchAsync	3189
6.642.2.18	setProducerId	3189
6.642.2.19	setWindowSize	3189
6.642.2.20	toString	3189
6.642.2.21	visit	3189
6.642.3	Field Documentation	3190
6.642.3.1	brokerPath	3190
6.642.3.2	destination	3190

6.642.3.3	dispatchAsync	3190
6.642.3.4	ID_PRODUCERINFO	3190
6.642.3.5	producerId	3190
6.642.3.6	windowSize	3190
6.643	activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller	
	Class Reference	3190
6.643.1	Detailed Description	3191
6.643.2	Constructor & Destructor Documentation	3191
6.643.2.1	ProducerInfoMarshaller	3191
6.643.2.2	~ProducerInfoMarshaller	3191
6.643.3	Member Function Documentation	3191
6.643.3.1	createObject	3191
6.643.3.2	getDataStructureType	3192
6.643.3.3	looseMarshal	3192
6.643.3.4	looseUnmarshal	3192
6.643.3.5	tightMarshal1	3193
6.643.3.6	tightMarshal2	3193
6.643.3.7	tightUnmarshal	3194
6.644	activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller	
	Class Reference	3194
6.644.1	Detailed Description	3195
6.644.2	Constructor & Destructor Documentation	3196
6.644.2.1	ProducerInfoMarshaller	3196
6.644.2.2	~ProducerInfoMarshaller	3196
6.644.3	Member Function Documentation	3196
6.644.3.1	createObject	3196
6.644.3.2	getDataStructureType	3196
6.644.3.3	looseMarshal	3196
6.644.3.4	looseUnmarshal	3197
6.644.3.5	tightMarshal1	3197
6.644.3.6	tightMarshal2	3198
6.644.3.7	tightUnmarshal	3198
6.645	activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	
	Class Reference	3199
6.645.1	Detailed Description	3200
6.645.2	Constructor & Destructor Documentation	3200
6.645.2.1	ProducerInfoMarshaller	3200
6.645.2.2	~ProducerInfoMarshaller	3200
6.645.3	Member Function Documentation	3200
6.645.3.1	createObject	3200
6.645.3.2	getDataStructureType	3200
6.645.3.3	looseMarshal	3200
6.645.3.4	looseUnmarshal	3201
6.645.3.5	tightMarshal1	3201
6.645.3.6	tightMarshal2	3202
6.645.3.7	tightUnmarshal	3202
6.646	activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller	
	Class Reference	3203
6.646.1	Detailed Description	3204
6.646.2	Constructor & Destructor Documentation	3204

6.646.2.1	ProducerInfoMarshaller . . . . .	3204
6.646.2.2	~ProducerInfoMarshaller . . . . .	3204
6.646.3	Member Function Documentation . . . . .	3204
6.646.3.1	createObject . . . . .	3204
6.646.3.2	getDataStructureType . . . . .	3204
6.646.3.3	looseMarshal . . . . .	3205
6.646.3.4	looseUnmarshal . . . . .	3205
6.646.3.5	tightMarshal1 . . . . .	3206
6.646.3.6	tightMarshal2 . . . . .	3206
6.646.3.7	tightUnmarshal . . . . .	3207
6.647	activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller	
	Class Reference . . . . .	3207
6.647.1	Detailed Description . . . . .	3208
6.647.2	Constructor & Destructor Documentation . . . . .	3208
6.647.2.1	ProducerInfoMarshaller . . . . .	3208
6.647.2.2	~ProducerInfoMarshaller . . . . .	3208
6.647.3	Member Function Documentation . . . . .	3208
6.647.3.1	createObject . . . . .	3208
6.647.3.2	getDataStructureType . . . . .	3209
6.647.3.3	looseMarshal . . . . .	3209
6.647.3.4	looseUnmarshal . . . . .	3209
6.647.3.5	tightMarshal1 . . . . .	3210
6.647.3.6	tightMarshal2 . . . . .	3210
6.647.3.7	tightUnmarshal . . . . .	3211
6.648	activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller	
	Class Reference . . . . .	3211
6.648.1	Detailed Description . . . . .	3212
6.648.2	Constructor & Destructor Documentation . . . . .	3213
6.648.2.1	ProducerInfoMarshaller . . . . .	3213
6.648.2.2	~ProducerInfoMarshaller . . . . .	3213
6.648.3	Member Function Documentation . . . . .	3213
6.648.3.1	createObject . . . . .	3213
6.648.3.2	getDataStructureType . . . . .	3213
6.648.3.3	looseMarshal . . . . .	3213
6.648.3.4	looseUnmarshal . . . . .	3214
6.648.3.5	tightMarshal1 . . . . .	3214
6.648.3.6	tightMarshal2 . . . . .	3215
6.648.3.7	tightUnmarshal . . . . .	3215
6.649	activemq::state::ProducerState Class Reference . . . . .	3216
6.649.1	Constructor & Destructor Documentation . . . . .	3216
6.649.1.1	ProducerState . . . . .	3216
6.649.1.2	~ProducerState . . . . .	3216
6.649.2	Member Function Documentation . . . . .	3216
6.649.2.1	getInfo . . . . .	3216
6.649.2.2	getTransactionState . . . . .	3216
6.649.2.3	setTransactionState . . . . .	3216
6.649.2.4	toString . . . . .	3216
6.650	decaf::util::Properties Class Reference . . . . .	3216
6.650.1	Detailed Description . . . . .	3218
6.650.2	Constructor & Destructor Documentation . . . . .	3219

6.650.2.1 Properties . . . . .	3219
6.650.2.2 Properties . . . . .	3219
6.650.2.3 ~Properties . . . . .	3219
6.650.3 Member Function Documentation . . . . .	3219
6.650.3.1 clear . . . . .	3219
6.650.3.2 clone . . . . .	3219
6.650.3.3 copy . . . . .	3219
6.650.3.4 equals . . . . .	3219
6.650.3.5 getProperty . . . . .	3220
6.650.3.6 getProperty . . . . .	3220
6.650.3.7 hasProperty . . . . .	3220
6.650.3.8 isEmpty . . . . .	3220
6.650.3.9 load . . . . .	3221
6.650.3.10load . . . . .	3221
6.650.3.11operator= . . . . .	3223
6.650.3.12propertyName . . . . .	3223
6.650.3.13remove . . . . .	3224
6.650.3.14setProperty . . . . .	3224
6.650.3.15size . . . . .	3224
6.650.3.16store . . . . .	3224
6.650.3.17store . . . . .	3225
6.650.3.18toArray . . . . .	3226
6.650.3.19toString . . . . .	3226
6.650.4 Field Documentation . . . . .	3226
6.650.4.1 defaults . . . . .	3226
6.651decaf::util::logging::PropertiesChangeListener Class Reference . . . .	3227
6.651.1 Detailed Description . . . . .	3227
6.651.2 Constructor & Destructor Documentation . . . . .	3227
6.651.2.1 ~PropertiesChangeListener . . . . .	3227
6.651.3 Member Function Documentation . . . . .	3227
6.651.3.1 onPropertiesReset . . . . .	3227
6.651.3.2 onPropertyChanged . . . . .	3228
6.652decaf::net::ProtocolException Class Reference . . . . .	3228
6.652.1 Constructor & Destructor Documentation . . . . .	3229
6.652.1.1 ProtocolException . . . . .	3229
6.652.1.2 ProtocolException . . . . .	3229
6.652.1.3 ProtocolException . . . . .	3229
6.652.1.4 ProtocolException . . . . .	3229
6.652.1.5 ProtocolException . . . . .	3230
6.652.1.6 ProtocolException . . . . .	3230
6.652.1.7 ~ProtocolException . . . . .	3230
6.652.2 Member Function Documentation . . . . .	3230
6.652.2.1 clone . . . . .	3230
6.653decaf::security::PublicKey Class Reference . . . . .	3231
6.653.1 Detailed Description . . . . .	3231
6.653.2 Constructor & Destructor Documentation . . . . .	3231
6.653.2.1 ~PublicKey . . . . .	3231
6.654decaf::io::PushbackInputStream Class Reference . . . . .	3231
6.654.1 Detailed Description . . . . .	3233
6.654.2 Constructor & Destructor Documentation . . . . .	3233

6.654.2.1	PushbackInputStream . . . . .	3233
6.654.2.2	PushbackInputStream . . . . .	3234
6.654.2.3	~PushbackInputStream . . . . .	3234
6.654.3	Member Function Documentation . . . . .	3234
6.654.3.1	available . . . . .	3234
6.654.3.2	doReadArrayBounded . . . . .	3235
6.654.3.3	doReadByte . . . . .	3235
6.654.3.4	mark . . . . .	3235
6.654.3.5	markSupported . . . . .	3235
6.654.3.6	reset . . . . .	3236
6.654.3.7	skip . . . . .	3236
6.654.3.8	unread . . . . .	3237
6.654.3.9	unread . . . . .	3237
6.654.3.10	unread . . . . .	3238
6.655	cms::Queue Class Reference . . . . .	3238
6.655.1	Detailed Description . . . . .	3239
6.655.2	Constructor & Destructor Documentation . . . . .	3239
6.655.2.1	~Queue . . . . .	3239
6.655.3	Member Function Documentation . . . . .	3239
6.655.3.1	getQueueName . . . . .	3239
6.656	decaf::util::Queue< E > Class Template Reference . . . . .	3239
6.656.1	Detailed Description . . . . .	3240
6.656.2	Constructor & Destructor Documentation . . . . .	3241
6.656.2.1	~Queue . . . . .	3241
6.656.3	Member Function Documentation . . . . .	3241
6.656.3.1	element . . . . .	3241
6.656.3.2	offer . . . . .	3241
6.656.3.3	peek . . . . .	3242
6.656.3.4	poll . . . . .	3242
6.656.3.5	remove . . . . .	3243
6.657	cms::QueueBrowser Class Reference . . . . .	3243
6.657.1	Detailed Description . . . . .	3244
6.657.2	Constructor & Destructor Documentation . . . . .	3244
6.657.2.1	~QueueBrowser . . . . .	3244
6.657.3	Member Function Documentation . . . . .	3244
6.657.3.1	getEnumeration . . . . .	3244
6.657.3.2	getMessageSelector . . . . .	3245
6.657.3.3	getQueue . . . . .	3245
6.658	decaf::util::Random Class Reference . . . . .	3245
6.658.1	Detailed Description . . . . .	3247
6.658.2	Constructor & Destructor Documentation . . . . .	3247
6.658.2.1	Random . . . . .	3247
6.658.2.2	Random . . . . .	3247
6.658.3	Member Function Documentation . . . . .	3247
6.658.3.1	next . . . . .	3247
6.658.3.2	nextBoolean . . . . .	3248
6.658.3.3	nextBytes . . . . .	3248
6.658.3.4	nextBytes . . . . .	3249
6.658.3.5	nextDouble . . . . .	3249
6.658.3.6	nextFloat . . . . .	3249

6.658.3.7 nextGaussian . . . . .	3249
6.658.3.8 nextInt . . . . .	3250
6.658.3.9 nextInt . . . . .	3250
6.658.3.10nextLong . . . . .	3250
6.658.3.11setSeed . . . . .	3251
6.659decaf::lang::Readable Class Reference . . . . .	3251
6.659.1 Detailed Description . . . . .	3252
6.659.2 Constructor & Destructor Documentation . . . . .	3252
6.659.2.1 ~Readable . . . . .	3252
6.659.3 Member Function Documentation . . . . .	3252
6.659.3.1 read . . . . .	3252
6.660activemq::transport::inactivity::ReadChecker Class Reference . . . . .	3253
6.660.1 Detailed Description . . . . .	3253
6.660.2 Constructor & Destructor Documentation . . . . .	3253
6.660.2.1 ReadChecker . . . . .	3253
6.660.2.2 ~ReadChecker . . . . .	3253
6.660.3 Member Function Documentation . . . . .	3253
6.660.3.1 run . . . . .	3253
6.661decaf::io::Reader Class Reference . . . . .	3254
6.661.1 Constructor & Destructor Documentation . . . . .	3255
6.661.1.1 Reader . . . . .	3255
6.661.1.2 ~Reader . . . . .	3255
6.661.2 Member Function Documentation . . . . .	3255
6.661.2.1 doReadArray . . . . .	3255
6.661.2.2 doReadArrayBounded . . . . .	3256
6.661.2.3 doReadChar . . . . .	3256
6.661.2.4 doReadCharBuffer . . . . .	3256
6.661.2.5 doReadVector . . . . .	3256
6.661.2.6 mark . . . . .	3256
6.661.2.7 markSupported . . . . .	3257
6.661.2.8 read . . . . .	3257
6.661.2.9 read . . . . .	3257
6.661.2.10read . . . . .	3258
6.661.2.11read . . . . .	3258
6.661.2.12read . . . . .	3259
6.661.2.13ready . . . . .	3259
6.661.2.14reset . . . . .	3260
6.661.2.15skip . . . . .	3260
6.662decaf::nio::ReadOnlyBufferException Class Reference . . . . .	3260
6.662.1 Constructor & Destructor Documentation . . . . .	3261
6.662.1.1 ReadOnlyBufferException . . . . .	3261
6.662.1.2 ReadOnlyBufferException . . . . .	3261
6.662.1.3 ReadOnlyBufferException . . . . .	3262
6.662.1.4 ReadOnlyBufferException . . . . .	3262
6.662.1.5 ReadOnlyBufferException . . . . .	3262
6.662.1.6 ReadOnlyBufferException . . . . .	3262
6.662.1.7 ~ReadOnlyBufferException . . . . .	3263
6.662.2 Member Function Documentation . . . . .	3263
6.662.2.1 clone . . . . .	3263
6.663decaf::util::concurrent::locks::ReadWriteLock Class Reference . . . . .	3263

6.663.1 Detailed Description . . . . .	3263
6.663.2 Constructor & Destructor Documentation . . . . .	3265
6.663.2.1 ~ReadWriteLock . . . . .	3265
6.663.3 Member Function Documentation . . . . .	3265
6.663.3.1 readLock . . . . .	3265
6.663.3.2 writeLock . . . . .	3265
6.664activemq::cmsutil::CmsTemplate::ReceiveExecutor Class Reference . . . . .	3265
6.664.1 Constructor & Destructor Documentation . . . . .	3266
6.664.1.1 ReceiveExecutor . . . . .	3266
6.664.1.2 ReceiveExecutor . . . . .	3266
6.664.1.3 ~ReceiveExecutor . . . . .	3266
6.664.2 Member Function Documentation . . . . .	3266
6.664.2.1 doInCms . . . . .	3266
6.664.2.2 getDestination . . . . .	3267
6.664.2.3 getMessage . . . . .	3267
6.664.2.4 operator= . . . . .	3267
6.664.3 Field Documentation . . . . .	3267
6.664.3.1 destination . . . . .	3267
6.664.3.2 message . . . . .	3267
6.664.3.3 noLocal . . . . .	3267
6.664.3.4 parent . . . . .	3267
6.664.3.5 selector . . . . .	3267
6.665activemq::core::RedeliveryPolicy Class Reference . . . . .	3267
6.665.1 Detailed Description . . . . .	3268
6.665.2 Constructor & Destructor Documentation . . . . .	3269
6.665.2.1 RedeliveryPolicy . . . . .	3269
6.665.2.2 ~RedeliveryPolicy . . . . .	3269
6.665.3 Member Function Documentation . . . . .	3269
6.665.3.1 clone . . . . .	3269
6.665.3.2 configure . . . . .	3269
6.665.3.3 getBackOffMultiplier . . . . .	3270
6.665.3.4 getCollisionAvoidancePercent . . . . .	3270
6.665.3.5 getInitialRedeliveryDelay . . . . .	3270
6.665.3.6 getMaximumRedeliveries . . . . .	3270
6.665.3.7 getRedeliveryDelay . . . . .	3270
6.665.3.8 isUseCollisionAvoidance . . . . .	3271
6.665.3.9 isUseExponentialBackOff . . . . .	3271
6.665.3.10setBackOffMultiplier . . . . .	3271
6.665.3.11setCollisionAvoidancePercent . . . . .	3271
6.665.3.12setInitialRedeliveryDelay . . . . .	3272
6.665.3.13setMaximumRedeliveries . . . . .	3272
6.665.3.14setUseCollisionAvoidance . . . . .	3272
6.665.3.15setUseExponentialBackOff . . . . .	3272
6.665.4 Field Documentation . . . . .	3272
6.665.4.1 NO_MAXIMUM_REDELIVERIES . . . . .	3272
6.666decaf::util::concurrent::locks::ReentrantLock Class Reference . . . . .	3273
6.666.1 Detailed Description . . . . .	3274
6.666.2 Constructor & Destructor Documentation . . . . .	3275
6.666.2.1 ReentrantLock . . . . .	3275
6.666.2.2 ~ReentrantLock . . . . .	3275

6.666.3 Member Function Documentation . . . . .	3275
6.666.3.1 getHoldCount . . . . .	3275
6.666.3.2 isFair . . . . .	3275
6.666.3.3 isHeldByCurrentThread . . . . .	3275
6.666.3.4 isLocked . . . . .	3276
6.666.3.5 lock . . . . .	3276
6.666.3.6 lockInterruptibly . . . . .	3276
6.666.3.7 newCondition . . . . .	3277
6.666.3.8 toString . . . . .	3278
6.666.3.9 tryLock . . . . .	3278
6.666.3.10tryLock . . . . .	3279
6.666.3.11unlock . . . . .	3280
6.667decaf::util::concurrent::RejectedExecutionException Class Reference . . . . .	3280
6.667.1 Constructor & Destructor Documentation . . . . .	3281
6.667.1.1 RejectedExecutionException . . . . .	3281
6.667.1.2 RejectedExecutionException . . . . .	3281
6.667.1.3 RejectedExecutionException . . . . .	3281
6.667.1.4 RejectedExecutionException . . . . .	3281
6.667.1.5 RejectedExecutionException . . . . .	3282
6.667.1.6 RejectedExecutionException . . . . .	3282
6.667.1.7 ~RejectedExecutionException . . . . .	3282
6.667.2 Member Function Documentation . . . . .	3282
6.667.2.1 clone . . . . .	3282
6.668decaf::util::concurrent::RejectedExecutionHandler Class Reference . . . . .	3283
6.668.1 Detailed Description . . . . .	3283
6.668.2 Constructor & Destructor Documentation . . . . .	3283
6.668.2.1 ~RejectedExecutionHandler . . . . .	3283
6.668.3 Member Function Documentation . . . . .	3283
6.668.3.1 rejectedExecution . . . . .	3283
6.669activemq::commands::RemoveInfo Class Reference . . . . .	3284
6.669.1 Constructor & Destructor Documentation . . . . .	3285
6.669.1.1 RemoveInfo . . . . .	3285
6.669.1.2 ~RemoveInfo . . . . .	3285
6.669.2 Member Function Documentation . . . . .	3285
6.669.2.1 cloneDataStructure . . . . .	3285
6.669.2.2 copyDataStructure . . . . .	3285
6.669.2.3 equals . . . . .	3286
6.669.2.4 getDataStructureType . . . . .	3286
6.669.2.5 getLastDeliveredSequenceId . . . . .	3286
6.669.2.6 getObjectId . . . . .	3286
6.669.2.7 getObjectId . . . . .	3286
6.669.2.8 isRemoveInfo . . . . .	3286
6.669.2.9 setLastDeliveredSequenceId . . . . .	3287
6.669.2.10setObjectId . . . . .	3287
6.669.2.11toString . . . . .	3287
6.669.2.12visit . . . . .	3287
6.669.3 Field Documentation . . . . .	3287
6.669.3.1 ID_REMOVEINFO . . . . .	3287
6.669.3.2 lastDeliveredSequenceId . . . . .	3287
6.669.3.3 objectId . . . . .	3287



6.670	activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller Class	
	Reference	3288
6.670.1	Detailed Description	3289
6.670.2	Constructor & Destructor Documentation	3289
6.670.2.1	RemoveInfoMarshaller	3289
6.670.2.2	~RemoveInfoMarshaller	3289
6.670.3	Member Function Documentation	3289
6.670.3.1	createObject	3289
6.670.3.2	getDataStructureType	3289
6.670.3.3	looseMarshal	3289
6.670.3.4	looseUnmarshal	3290
6.670.3.5	tightMarshal1	3290
6.670.3.6	tightMarshal2	3291
6.670.3.7	tightUnmarshal	3291
6.671	activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller Class	
	Reference	3292
6.671.1	Detailed Description	3293
6.671.2	Constructor & Destructor Documentation	3293
6.671.2.1	RemoveInfoMarshaller	3293
6.671.2.2	~RemoveInfoMarshaller	3293
6.671.3	Member Function Documentation	3293
6.671.3.1	createObject	3293
6.671.3.2	getDataStructureType	3293
6.671.3.3	looseMarshal	3294
6.671.3.4	looseUnmarshal	3294
6.671.3.5	tightMarshal1	3295
6.671.3.6	tightMarshal2	3295
6.671.3.7	tightUnmarshal	3296
6.672	activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller Class	
	Reference	3296
6.672.1	Detailed Description	3297
6.672.2	Constructor & Destructor Documentation	3297
6.672.2.1	RemoveInfoMarshaller	3297
6.672.2.2	~RemoveInfoMarshaller	3297
6.672.3	Member Function Documentation	3297
6.672.3.1	createObject	3297
6.672.3.2	getDataStructureType	3298
6.672.3.3	looseMarshal	3298
6.672.3.4	looseUnmarshal	3298
6.672.3.5	tightMarshal1	3299
6.672.3.6	tightMarshal2	3299
6.672.3.7	tightUnmarshal	3300
6.673	activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller Class	
	Reference	3300
6.673.1	Detailed Description	3301
6.673.2	Constructor & Destructor Documentation	3302
6.673.2.1	RemoveInfoMarshaller	3302
6.673.2.2	~RemoveInfoMarshaller	3302
6.673.3	Member Function Documentation	3302
6.673.3.1	createObject	3302

6.673.3.2	getDataStructureType	3302
6.673.3.3	looseMarshal	3302
6.673.3.4	looseUnmarshal	3303
6.673.3.5	tightMarshal1	3303
6.673.3.6	tightMarshal2	3304
6.673.3.7	tightUnmarshal	3304
6.674	activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller Class	
	Reference	3305
6.674.1	Detailed Description	3306
6.674.2	Constructor & Destructor Documentation	3306
6.674.2.1	RemoveInfoMarshaller	3306
6.674.2.2	~RemoveInfoMarshaller	3306
6.674.3	Member Function Documentation	3306
6.674.3.1	createObject	3306
6.674.3.2	getDataStructureType	3306
6.674.3.3	looseMarshal	3306
6.674.3.4	looseUnmarshal	3307
6.674.3.5	tightMarshal1	3307
6.674.3.6	tightMarshal2	3308
6.674.3.7	tightUnmarshal	3308
6.675	activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller Class	
	Reference	3309
6.675.1	Detailed Description	3310
6.675.2	Constructor & Destructor Documentation	3310
6.675.2.1	RemoveInfoMarshaller	3310
6.675.2.2	~RemoveInfoMarshaller	3310
6.675.3	Member Function Documentation	3310
6.675.3.1	createObject	3310
6.675.3.2	getDataStructureType	3310
6.675.3.3	looseMarshal	3311
6.675.3.4	looseUnmarshal	3311
6.675.3.5	tightMarshal1	3312
6.675.3.6	tightMarshal2	3312
6.675.3.7	tightUnmarshal	3313
6.676	activemq::commands::RemoveSubscriptionInfo Class Reference	3313
6.676.1	Constructor & Destructor Documentation	3315
6.676.1.1	RemoveSubscriptionInfo	3315
6.676.1.2	~RemoveSubscriptionInfo	3315
6.676.2	Member Function Documentation	3315
6.676.2.1	cloneDataStructure	3315
6.676.2.2	copyDataStructure	3315
6.676.2.3	equals	3315
6.676.2.4	getClientId	3316
6.676.2.5	getClientId	3316
6.676.2.6	getConnectionId	3316
6.676.2.7	getConnectionId	3316
6.676.2.8	getDataStructureType	3316
6.676.2.9	getSubscriptionName	3316
6.676.2.10	getSubscriptionName	3316
6.676.2.11	isRemoveSubscriptionInfo	3316

6.676.2.12	setClientId	3317
6.676.2.13	setConnectionId	3317
6.676.2.14	setSubscriptionName	3317
6.676.2.15	toString	3317
6.676.2.16	visit	3317
6.676.3	Field Documentation	3318
6.676.3.1	clientId	3318
6.676.3.2	connectionId	3318
6.676.3.3	ID_REMOVE SUBSCRIPTION INFO	3318
6.676.3.4	subscriptionName	3318
6.677	activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller	
	Class Reference	3318
6.677.1	Detailed Description	3319
6.677.2	Constructor & Destructor Documentation	3319
6.677.2.1	RemoveSubscriptionInfoMarshaller	3319
6.677.2.2	~RemoveSubscriptionInfoMarshaller	3319
6.677.3	Member Function Documentation	3319
6.677.3.1	createObject	3319
6.677.3.2	getDataStructureType	3320
6.677.3.3	looseMarshal	3320
6.677.3.4	looseUnmarshal	3320
6.677.3.5	tightMarshal1	3321
6.677.3.6	tightMarshal2	3321
6.677.3.7	tightUnmarshal	3322
6.678	activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller	
	Class Reference	3322
6.678.1	Detailed Description	3323
6.678.2	Constructor & Destructor Documentation	3323
6.678.2.1	RemoveSubscriptionInfoMarshaller	3323
6.678.2.2	~RemoveSubscriptionInfoMarshaller	3323
6.678.3	Member Function Documentation	3323
6.678.3.1	createObject	3323
6.678.3.2	getDataStructureType	3324
6.678.3.3	looseMarshal	3324
6.678.3.4	looseUnmarshal	3324
6.678.3.5	tightMarshal1	3325
6.678.3.6	tightMarshal2	3325
6.678.3.7	tightUnmarshal	3326
6.679	activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller	
	Class Reference	3326
6.679.1	Detailed Description	3327
6.679.2	Constructor & Destructor Documentation	3328
6.679.2.1	RemoveSubscriptionInfoMarshaller	3328
6.679.2.2	~RemoveSubscriptionInfoMarshaller	3328
6.679.3	Member Function Documentation	3328
6.679.3.1	createObject	3328
6.679.3.2	getDataStructureType	3328
6.679.3.3	looseMarshal	3328
6.679.3.4	looseUnmarshal	3329
6.679.3.5	tightMarshal1	3329

6.679.3.6	tightMarshal2	3330
6.679.3.7	tightUnmarshal	3330
6.680	activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller	
	Class Reference	3331
6.680.1	Detailed Description	3332
6.680.2	Constructor & Destructor Documentation	3332
6.680.2.1	RemoveSubscriptionInfoMarshaller	3332
6.680.2.2	~RemoveSubscriptionInfoMarshaller	3332
6.680.3	Member Function Documentation	3332
6.680.3.1	createObject	3332
6.680.3.2	getDataStructureType	3332
6.680.3.3	looseMarshal	3332
6.680.3.4	looseUnmarshal	3333
6.680.3.5	tightMarshal1	3333
6.680.3.6	tightMarshal2	3334
6.680.3.7	tightUnmarshal	3334
6.681	activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller	
	Class Reference	3335
6.681.1	Detailed Description	3336
6.681.2	Constructor & Destructor Documentation	3336
6.681.2.1	RemoveSubscriptionInfoMarshaller	3336
6.681.2.2	~RemoveSubscriptionInfoMarshaller	3336
6.681.3	Member Function Documentation	3336
6.681.3.1	createObject	3336
6.681.3.2	getDataStructureType	3336
6.681.3.3	looseMarshal	3337
6.681.3.4	looseUnmarshal	3337
6.681.3.5	tightMarshal1	3338
6.681.3.6	tightMarshal2	3338
6.681.3.7	tightUnmarshal	3339
6.682	activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller	
	Class Reference	3339
6.682.1	Detailed Description	3340
6.682.2	Constructor & Destructor Documentation	3340
6.682.2.1	RemoveSubscriptionInfoMarshaller	3340
6.682.2.2	~RemoveSubscriptionInfoMarshaller	3340
6.682.3	Member Function Documentation	3340
6.682.3.1	createObject	3340
6.682.3.2	getDataStructureType	3341
6.682.3.3	looseMarshal	3341
6.682.3.4	looseUnmarshal	3341
6.682.3.5	tightMarshal1	3342
6.682.3.6	tightMarshal2	3342
6.682.3.7	tightUnmarshal	3343
6.683	activemq::commands::ReplayCommand Class Reference	3343
6.683.1	Constructor & Destructor Documentation	3345
6.683.1.1	ReplayCommand	3345
6.683.1.2	~ReplayCommand	3345
6.683.2	Member Function Documentation	3345
6.683.2.1	cloneDataStructure	3345

6.683.2.2	copyDataStructure . . . . .	3345
6.683.2.3	equals . . . . .	3345
6.683.2.4	getDataStructureType . . . . .	3346
6.683.2.5	getFirstNakNumber . . . . .	3346
6.683.2.6	getLastNakNumber . . . . .	3346
6.683.2.7	setFirstNakNumber . . . . .	3346
6.683.2.8	setLastNakNumber . . . . .	3346
6.683.2.9	toString . . . . .	3346
6.683.2.10	visit . . . . .	3346
6.683.3	Field Documentation . . . . .	3347
6.683.3.1	firstNakNumber . . . . .	3347
6.683.3.2	ID_REPLAYCOMMAND . . . . .	3347
6.683.3.3	lastNakNumber . . . . .	3347
6.684	activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller	
	Class Reference . . . . .	3347
6.684.1	Detailed Description . . . . .	3348
6.684.2	Constructor & Destructor Documentation . . . . .	3348
6.684.2.1	ReplayCommandMarshaller . . . . .	3348
6.684.2.2	~ReplayCommandMarshaller . . . . .	3348
6.684.3	Member Function Documentation . . . . .	3348
6.684.3.1	createObject . . . . .	3348
6.684.3.2	getDataStructureType . . . . .	3348
6.684.3.3	looseMarshal . . . . .	3349
6.684.3.4	looseUnmarshal . . . . .	3349
6.684.3.5	tightMarshal1 . . . . .	3350
6.684.3.6	tightMarshal2 . . . . .	3350
6.684.3.7	tightUnmarshal . . . . .	3351
6.685	activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller	
	Class Reference . . . . .	3351
6.685.1	Detailed Description . . . . .	3352
6.685.2	Constructor & Destructor Documentation . . . . .	3352
6.685.2.1	ReplayCommandMarshaller . . . . .	3352
6.685.2.2	~ReplayCommandMarshaller . . . . .	3352
6.685.3	Member Function Documentation . . . . .	3352
6.685.3.1	createObject . . . . .	3352
6.685.3.2	getDataStructureType . . . . .	3353
6.685.3.3	looseMarshal . . . . .	3353
6.685.3.4	looseUnmarshal . . . . .	3353
6.685.3.5	tightMarshal1 . . . . .	3354
6.685.3.6	tightMarshal2 . . . . .	3354
6.685.3.7	tightUnmarshal . . . . .	3355
6.686	activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller	
	Class Reference . . . . .	3355
6.686.1	Detailed Description . . . . .	3356
6.686.2	Constructor & Destructor Documentation . . . . .	3357
6.686.2.1	ReplayCommandMarshaller . . . . .	3357
6.686.2.2	~ReplayCommandMarshaller . . . . .	3357
6.686.3	Member Function Documentation . . . . .	3357
6.686.3.1	createObject . . . . .	3357
6.686.3.2	getDataStructureType . . . . .	3357

6.686.3.3	looseMarshal . . . . .	3357
6.686.3.4	looseUnmarshal . . . . .	3358
6.686.3.5	tightMarshal1 . . . . .	3358
6.686.3.6	tightMarshal2 . . . . .	3359
6.686.3.7	tightUnmarshal . . . . .	3359
6.687	activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller	
	Class Reference . . . . .	3360
6.687.1	Detailed Description . . . . .	3361
6.687.2	Constructor & Destructor Documentation . . . . .	3361
6.687.2.1	ReplayCommandMarshaller . . . . .	3361
6.687.2.2	~ReplayCommandMarshaller . . . . .	3361
6.687.3	Member Function Documentation . . . . .	3361
6.687.3.1	createObject . . . . .	3361
6.687.3.2	getDataStructureType . . . . .	3361
6.687.3.3	looseMarshal . . . . .	3361
6.687.3.4	looseUnmarshal . . . . .	3362
6.687.3.5	tightMarshal1 . . . . .	3362
6.687.3.6	tightMarshal2 . . . . .	3363
6.687.3.7	tightUnmarshal . . . . .	3363
6.688	activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller	
	Class Reference . . . . .	3364
6.688.1	Detailed Description . . . . .	3365
6.688.2	Constructor & Destructor Documentation . . . . .	3365
6.688.2.1	ReplayCommandMarshaller . . . . .	3365
6.688.2.2	~ReplayCommandMarshaller . . . . .	3365
6.688.3	Member Function Documentation . . . . .	3365
6.688.3.1	createObject . . . . .	3365
6.688.3.2	getDataStructureType . . . . .	3365
6.688.3.3	looseMarshal . . . . .	3366
6.688.3.4	looseUnmarshal . . . . .	3366
6.688.3.5	tightMarshal1 . . . . .	3367
6.688.3.6	tightMarshal2 . . . . .	3367
6.688.3.7	tightUnmarshal . . . . .	3368
6.689	activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller	
	Class Reference . . . . .	3368
6.689.1	Detailed Description . . . . .	3369
6.689.2	Constructor & Destructor Documentation . . . . .	3369
6.689.2.1	ReplayCommandMarshaller . . . . .	3369
6.689.2.2	~ReplayCommandMarshaller . . . . .	3369
6.689.3	Member Function Documentation . . . . .	3369
6.689.3.1	createObject . . . . .	3369
6.689.3.2	getDataStructureType . . . . .	3370
6.689.3.3	looseMarshal . . . . .	3370
6.689.3.4	looseUnmarshal . . . . .	3370
6.689.3.5	tightMarshal1 . . . . .	3371
6.689.3.6	tightMarshal2 . . . . .	3371
6.689.3.7	tightUnmarshal . . . . .	3372
6.690	activemq::cmsutil::CmsTemplate::ResolveProducerExecutor Class Reference . . . . .	3372
6.690.1	Constructor & Destructor Documentation . . . . .	3373

6.690.1.1	ResolveProducerExecutor . . . . .	3373
6.690.1.2	~ResolveProducerExecutor . . . . .	3373
6.690.2	Member Function Documentation . . . . .	3373
6.690.2.1	getDestination . . . . .	3373
6.690.2.2	operator= . . . . .	3373
6.691	activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor Class Reference . . . . .	3373
6.691.1	Constructor & Destructor Documentation . . . . .	3374
6.691.1.1	ResolveReceiveExecutor . . . . .	3374
6.691.1.2	~ResolveReceiveExecutor . . . . .	3374
6.691.2	Member Function Documentation . . . . .	3374
6.691.2.1	getDestination . . . . .	3374
6.691.2.2	operator= . . . . .	3374
6.692	decaf::internal::util::Resource Class Reference . . . . .	3374
6.692.1	Detailed Description . . . . .	3375
6.692.2	Constructor & Destructor Documentation . . . . .	3375
6.692.2.1	~Resource . . . . .	3375
6.693	decaf::internal::util::ResourceLifecycleManager Class Reference . . . . .	3375
6.693.1	Detailed Description . . . . .	3375
6.693.2	Constructor & Destructor Documentation . . . . .	3376
6.693.2.1	ResourceLifecycleManager . . . . .	3376
6.693.2.2	~ResourceLifecycleManager . . . . .	3376
6.693.3	Member Function Documentation . . . . .	3376
6.693.3.1	addResource . . . . .	3376
6.693.3.2	destroyResources . . . . .	3376
6.694	activemq::cmsutil::ResourceLifecycleManager Class Reference . . . . .	3376
6.694.1	Detailed Description . . . . .	3377
6.694.2	Constructor & Destructor Documentation . . . . .	3377
6.694.2.1	ResourceLifecycleManager . . . . .	3377
6.694.2.2	ResourceLifecycleManager . . . . .	3377
6.694.2.3	~ResourceLifecycleManager . . . . .	3377
6.694.3	Member Function Documentation . . . . .	3377
6.694.3.1	addConnection . . . . .	3377
6.694.3.2	addDestination . . . . .	3378
6.694.3.3	addMessageConsumer . . . . .	3378
6.694.3.4	addMessageProducer . . . . .	3378
6.694.3.5	addSession . . . . .	3378
6.694.3.6	destroy . . . . .	3378
6.694.3.7	operator= . . . . .	3379
6.694.3.8	releaseAll . . . . .	3379
6.695	activemq::commands::Response Class Reference . . . . .	3379
6.695.1	Constructor & Destructor Documentation . . . . .	3380
6.695.1.1	Response . . . . .	3380
6.695.1.2	~Response . . . . .	3380
6.695.2	Member Function Documentation . . . . .	3380
6.695.2.1	cloneDataStructure . . . . .	3380
6.695.2.2	copyDataStructure . . . . .	3380
6.695.2.3	equals . . . . .	3381
6.695.2.4	getCorrelationId . . . . .	3381
6.695.2.5	getDataStructureType . . . . .	3381

6.695.2.6	isResponse . . . . .	3381
6.695.2.7	setCorrelationId . . . . .	3382
6.695.2.8	toString . . . . .	3382
6.695.2.9	visit . . . . .	3382
6.695.3	Field Documentation . . . . .	3382
6.695.3.1	correlationId . . . . .	3382
6.695.3.2	ID_RESPONSE . . . . .	3382
6.696	activemq::transport::mock::ResponseBuilder Class Reference . . . . .	3382
6.696.1	Detailed Description . . . . .	3383
6.696.2	Constructor & Destructor Documentation . . . . .	3383
6.696.2.1	~ResponseBuilder . . . . .	3383
6.696.3	Member Function Documentation . . . . .	3383
6.696.3.1	buildIncomingCommands . . . . .	3383
6.696.3.2	buildResponse . . . . .	3384
6.697	activemq::transport::correlator::ResponseCorrelator Class Reference . . . . .	3384
6.697.1	Detailed Description . . . . .	3385
6.697.2	Constructor & Destructor Documentation . . . . .	3385
6.697.2.1	ResponseCorrelator . . . . .	3385
6.697.2.2	~ResponseCorrelator . . . . .	3386
6.697.3	Member Function Documentation . . . . .	3386
6.697.3.1	close . . . . .	3386
6.697.3.2	onCommand . . . . .	3386
6.697.3.3	oneway . . . . .	3386
6.697.3.4	onTransportException . . . . .	3387
6.697.3.5	request . . . . .	3387
6.697.3.6	request . . . . .	3387
6.697.3.7	start . . . . .	3388
6.698	activemq::wireformat::openwire::marshal::v4::ResponseMarshaller Class Reference . . . . .	3388
6.698.1	Detailed Description . . . . .	3389
6.698.2	Constructor & Destructor Documentation . . . . .	3390
6.698.2.1	ResponseMarshaller . . . . .	3390
6.698.2.2	~ResponseMarshaller . . . . .	3390
6.698.3	Member Function Documentation . . . . .	3390
6.698.3.1	createObject . . . . .	3390
6.698.3.2	getDataStructureType . . . . .	3390
6.698.3.3	looseMarshal . . . . .	3390
6.698.3.4	looseUnmarshal . . . . .	3391
6.698.3.5	tightMarshal1 . . . . .	3392
6.698.3.6	tightMarshal2 . . . . .	3392
6.698.3.7	tightUnmarshal . . . . .	3393
6.699	activemq::wireformat::openwire::marshal::v2::ResponseMarshaller Class Reference . . . . .	3393
6.699.1	Detailed Description . . . . .	3394
6.699.2	Constructor & Destructor Documentation . . . . .	3395
6.699.2.1	ResponseMarshaller . . . . .	3395
6.699.2.2	~ResponseMarshaller . . . . .	3395
6.699.3	Member Function Documentation . . . . .	3395
6.699.3.1	createObject . . . . .	3395
6.699.3.2	getDataStructureType . . . . .	3395



6.699.3.3 looseMarshal . . . . .	3395
6.699.3.4 looseUnmarshal . . . . .	3396
6.699.3.5 tightMarshal1 . . . . .	3397
6.699.3.6 tightMarshal2 . . . . .	3397
6.699.3.7 tightUnmarshal . . . . .	3398
6.700activemq::wireformat::openwire::marshal::v5::ResponseMarshaller Class	
Reference . . . . .	3398
6.700.1 Detailed Description . . . . .	3399
6.700.2 Constructor & Destructor Documentation . . . . .	3400
6.700.2.1 ResponseMarshaller . . . . .	3400
6.700.2.2 ~ResponseMarshaller . . . . .	3400
6.700.3 Member Function Documentation . . . . .	3400
6.700.3.1 createObject . . . . .	3400
6.700.3.2 getDataStructureType . . . . .	3400
6.700.3.3 looseMarshal . . . . .	3400
6.700.3.4 looseUnmarshal . . . . .	3401
6.700.3.5 tightMarshal1 . . . . .	3402
6.700.3.6 tightMarshal2 . . . . .	3402
6.700.3.7 tightUnmarshal . . . . .	3403
6.701activemq::wireformat::openwire::marshal::v3::ResponseMarshaller Class	
Reference . . . . .	3403
6.701.1 Detailed Description . . . . .	3404
6.701.2 Constructor & Destructor Documentation . . . . .	3405
6.701.2.1 ResponseMarshaller . . . . .	3405
6.701.2.2 ~ResponseMarshaller . . . . .	3405
6.701.3 Member Function Documentation . . . . .	3405
6.701.3.1 createObject . . . . .	3405
6.701.3.2 getDataStructureType . . . . .	3405
6.701.3.3 looseMarshal . . . . .	3405
6.701.3.4 looseUnmarshal . . . . .	3406
6.701.3.5 tightMarshal1 . . . . .	3407
6.701.3.6 tightMarshal2 . . . . .	3407
6.701.3.7 tightUnmarshal . . . . .	3408
6.702activemq::wireformat::openwire::marshal::v1::ResponseMarshaller Class	
Reference . . . . .	3408
6.702.1 Detailed Description . . . . .	3409
6.702.2 Constructor & Destructor Documentation . . . . .	3410
6.702.2.1 ResponseMarshaller . . . . .	3410
6.702.2.2 ~ResponseMarshaller . . . . .	3410
6.702.3 Member Function Documentation . . . . .	3410
6.702.3.1 createObject . . . . .	3410
6.702.3.2 getDataStructureType . . . . .	3410
6.702.3.3 looseMarshal . . . . .	3410
6.702.3.4 looseUnmarshal . . . . .	3411
6.702.3.5 tightMarshal1 . . . . .	3412
6.702.3.6 tightMarshal2 . . . . .	3412
6.702.3.7 tightUnmarshal . . . . .	3413
6.703activemq::wireformat::openwire::marshal::v6::ResponseMarshaller Class	
Reference . . . . .	3413
6.703.1 Detailed Description . . . . .	3414

6.703.2 Constructor & Destructor Documentation . . . . .	3415
6.703.2.1 ResponseMarshaller . . . . .	3415
6.703.2.2 ~ResponseMarshaller . . . . .	3415
6.703.3 Member Function Documentation . . . . .	3415
6.703.3.1 createObject . . . . .	3415
6.703.3.2 getDataStructureType . . . . .	3415
6.703.3.3 looseMarshal . . . . .	3415
6.703.3.4 looseUnmarshal . . . . .	3416
6.703.3.5 tightMarshal1 . . . . .	3417
6.703.3.6 tightMarshal2 . . . . .	3417
6.703.3.7 tightUnmarshal . . . . .	3418
6.704decaf::lang::Runnable Class Reference . . . . .	3418
6.704.1 Detailed Description . . . . .	3419
6.704.2 Constructor & Destructor Documentation . . . . .	3419
6.704.2.1 ~Runnable . . . . .	3419
6.704.3 Member Function Documentation . . . . .	3419
6.704.3.1 run . . . . .	3419
6.705decaf::lang::Runtime Class Reference . . . . .	3419
6.705.1 Constructor & Destructor Documentation . . . . .	3420
6.705.1.1 ~Runtime . . . . .	3420
6.705.2 Member Function Documentation . . . . .	3420
6.705.2.1 getRuntime . . . . .	3420
6.705.2.2 initializeRuntime . . . . .	3420
6.705.2.3 initializeRuntime . . . . .	3420
6.705.2.4 shutdownRuntime . . . . .	3421
6.706decaf::lang::exceptions::RuntimeException Class Reference . . . . .	3421
6.706.1 Constructor & Destructor Documentation . . . . .	3422
6.706.1.1 RuntimeException . . . . .	3422
6.706.1.2 RuntimeException . . . . .	3422
6.706.1.3 RuntimeException . . . . .	3422
6.706.1.4 RuntimeException . . . . .	3422
6.706.1.5 RuntimeException . . . . .	3423
6.706.1.6 RuntimeException . . . . .	3423
6.706.1.7 ~RuntimeException . . . . .	3423
6.706.2 Member Function Documentation . . . . .	3423
6.706.2.1 clone . . . . .	3423
6.707decaf::security::SecureRandom Class Reference . . . . .	3424
6.707.1 Detailed Description . . . . .	3426
6.707.2 Constructor & Destructor Documentation . . . . .	3426
6.707.2.1 SecureRandom . . . . .	3426
6.707.2.2 SecureRandom . . . . .	3426
6.707.2.3 SecureRandom . . . . .	3426
6.707.2.4 ~SecureRandom . . . . .	3427
6.707.3 Member Function Documentation . . . . .	3427
6.707.3.1 next . . . . .	3427
6.707.3.2 nextBytes . . . . .	3427
6.707.3.3 nextBytes . . . . .	3428
6.707.3.4 setSeed . . . . .	3428
6.707.3.5 setSeed . . . . .	3428
6.707.3.6 setSeed . . . . .	3429

6.708	decaf::internal::security::SecureRandomImpl Class Reference	3429
6.708.1	Detailed Description	3430
6.708.2	Constructor & Destructor Documentation	3430
6.708.2.1	SecureRandomImpl	3430
6.708.2.2	~SecureRandomImpl	3430
6.708.2.3	SecureRandomImpl	3430
6.708.2.4	~SecureRandomImpl	3430
6.708.3	Member Function Documentation	3430
6.708.3.1	providerGenerateSeed	3430
6.708.3.2	providerGenerateSeed	3431
6.708.3.3	providerNextBytes	3431
6.708.3.4	providerNextBytes	3431
6.708.3.5	providerSetSeed	3432
6.708.3.6	providerSetSeed	3432
6.709	decaf::security::SecureRandomSpi Class Reference	3432
6.709.1	Detailed Description	3433
6.709.2	Constructor & Destructor Documentation	3433
6.709.2.1	SecureRandomSpi	3433
6.709.2.2	~SecureRandomSpi	3433
6.709.3	Member Function Documentation	3433
6.709.3.1	providerGenerateSeed	3433
6.709.3.2	providerNextBytes	3434
6.709.3.3	providerSetSeed	3434
6.710	decaf::util::concurrent::Semaphore Class Reference	3434
6.710.1	Detailed Description	3436
6.710.2	Constructor & Destructor Documentation	3437
6.710.2.1	Semaphore	3437
6.710.2.2	Semaphore	3438
6.710.2.3	~Semaphore	3438
6.710.3	Member Function Documentation	3438
6.710.3.1	acquire	3438
6.710.3.2	acquire	3439
6.710.3.3	acquireUninterruptibly	3439
6.710.3.4	acquireUninterruptibly	3440
6.710.3.5	availablePermits	3440
6.710.3.6	drainPermits	3441
6.710.3.7	isFair	3441
6.710.3.8	release	3441
6.710.3.9	release	3441
6.710.3.10	toString	3442
6.710.3.11	tryAcquire	3442
6.710.3.12	tryAcquire	3443
6.710.3.13	tryAcquire	3444
6.710.3.14	tryAcquire	3445
6.711	activemq::cmsutil::CmsTemplate::SendExecutor Class Reference	3445
6.711.1	Constructor & Destructor Documentation	3446
6.711.1.1	SendExecutor	3446
6.711.1.2	SendExecutor	3446
6.711.1.3	~SendExecutor	3446
6.711.2	Member Function Documentation	3446

6.711.2.1 doInCms . . . . .	3446
6.711.2.2 operator= . . . . .	3447
6.712decaf::net::ServerSocket Class Reference . . . . .	3447
6.712.1 Detailed Description . . . . .	3449
6.712.2 Constructor & Destructor Documentation . . . . .	3449
6.712.2.1 ServerSocket . . . . .	3449
6.712.2.2 ServerSocket . . . . .	3449
6.712.2.3 ServerSocket . . . . .	3450
6.712.2.4 ServerSocket . . . . .	3450
6.712.2.5 ~ServerSocket . . . . .	3451
6.712.2.6 ServerSocket . . . . .	3451
6.712.3 Member Function Documentation . . . . .	3451
6.712.3.1 accept . . . . .	3451
6.712.3.2 bind . . . . .	3452
6.712.3.3 bind . . . . .	3452
6.712.3.4 checkClosed . . . . .	3453
6.712.3.5 close . . . . .	3453
6.712.3.6 ensureCreated . . . . .	3453
6.712.3.7 getDefaultBacklog . . . . .	3453
6.712.3.8 getLocalPort . . . . .	3453
6.712.3.9 getReceiveBufferSize . . . . .	3453
6.712.3.10getReuseAddress . . . . .	3454
6.712.3.11getSoTimeout . . . . .	3454
6.712.3.12implAccept . . . . .	3454
6.712.3.13isBound . . . . .	3455
6.712.3.14isClosed . . . . .	3455
6.712.3.15setReceiveBufferSize . . . . .	3455
6.712.3.16setReuseAddress . . . . .	3455
6.712.3.17setSocketImplFactory . . . . .	3456
6.712.3.18setSoTimeout . . . . .	3456
6.712.3.19setupSocketImpl . . . . .	3456
6.712.3.20toString . . . . .	3456
6.713decaf::net::ServerSocketFactory Class Reference . . . . .	3457
6.713.1 Detailed Description . . . . .	3457
6.713.2 Constructor & Destructor Documentation . . . . .	3458
6.713.2.1 ServerSocketFactory . . . . .	3458
6.713.2.2 ~ServerSocketFactory . . . . .	3458
6.713.3 Member Function Documentation . . . . .	3458
6.713.3.1 createServerSocket . . . . .	3458
6.713.3.2 createServerSocket . . . . .	3458
6.713.3.3 createServerSocket . . . . .	3459
6.713.3.4 createServerSocket . . . . .	3459
6.713.3.5 getDefault . . . . .	3460
6.714cms::Session Class Reference . . . . .	3460
6.714.1 Detailed Description . . . . .	3463
6.714.2 Member Enumeration Documentation . . . . .	3464
6.714.2.1 AcknowledgeMode . . . . .	3464
6.714.3 Constructor & Destructor Documentation . . . . .	3464
6.714.3.1 ~Session . . . . .	3464
6.714.4 Member Function Documentation . . . . .	3464

6.714.4.1 close . . . . .	3464
6.714.4.2 commit . . . . .	3465
6.714.4.3 createBrowser . . . . .	3465
6.714.4.4 createBrowser . . . . .	3466
6.714.4.5 createBytesMessage . . . . .	3466
6.714.4.6 createBytesMessage . . . . .	3466
6.714.4.7 createConsumer . . . . .	3467
6.714.4.8 createConsumer . . . . .	3467
6.714.4.9 createConsumer . . . . .	3468
6.714.4.10createDurableConsumer . . . . .	3469
6.714.4.11createMapMessage . . . . .	3469
6.714.4.12createMessage . . . . .	3470
6.714.4.13createProducer . . . . .	3470
6.714.4.14createQueue . . . . .	3470
6.714.4.15createStreamMessage . . . . .	3471
6.714.4.16createTemporaryQueue . . . . .	3471
6.714.4.17createTemporaryTopic . . . . .	3471
6.714.4.18createTextMessage . . . . .	3472
6.714.4.19createTextMessage . . . . .	3472
6.714.4.20createTopic . . . . .	3472
6.714.4.21getAcknowledgeMode . . . . .	3473
6.714.4.22isTransacted . . . . .	3473
6.714.4.23recover . . . . .	3474
6.714.4.24rollback . . . . .	3474
6.714.4.25unsubscribe . . . . .	3475
6.715activemq::cmsutil::SessionCallback Class Reference . . . . .	3475
6.715.1 Detailed Description . . . . .	3475
6.715.2 Constructor & Destructor Documentation . . . . .	3476
6.715.2.1 ~SessionCallback . . . . .	3476
6.715.3 Member Function Documentation . . . . .	3476
6.715.3.1 doInCms . . . . .	3476
6.716activemq::commands::SessionId Class Reference . . . . .	3476
6.716.1 Member Typedef Documentation . . . . .	3478
6.716.1.1 COMPARATOR . . . . .	3478
6.716.2 Constructor & Destructor Documentation . . . . .	3478
6.716.2.1 SessionId . . . . .	3478
6.716.2.2 SessionId . . . . .	3478
6.716.2.3 SessionId . . . . .	3478
6.716.2.4 SessionId . . . . .	3478
6.716.2.5 SessionId . . . . .	3478
6.716.2.6 ~SessionId . . . . .	3478
6.716.3 Member Function Documentation . . . . .	3478
6.716.3.1 cloneDataStructure . . . . .	3478
6.716.3.2 compareTo . . . . .	3478
6.716.3.3 copyDataStructure . . . . .	3478
6.716.3.4 equals . . . . .	3479
6.716.3.5 equals . . . . .	3479
6.716.3.6 getConnectionId . . . . .	3479
6.716.3.7 getConnectionId . . . . .	3479
6.716.3.8 getDataStructureType . . . . .	3479

6.716.3.9	getParentId . . . . .	3480
6.716.3.10	getValue . . . . .	3480
6.716.3.11	operator< . . . . .	3480
6.716.3.12	operator= . . . . .	3480
6.716.3.13	operator== . . . . .	3480
6.716.3.14	setConnectionId . . . . .	3480
6.716.3.15	setValue . . . . .	3480
6.716.3.16	toString . . . . .	3480
6.716.4	Field Documentation . . . . .	3480
6.716.4.1	connectionId . . . . .	3480
6.716.4.2	ID_SESSIONID . . . . .	3480
6.716.4.3	value . . . . .	3480
6.717	activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller Class	
	Reference . . . . .	3481
6.717.1	Detailed Description . . . . .	3482
6.717.2	Constructor & Destructor Documentation . . . . .	3482
6.717.2.1	SessionIdMarshaller . . . . .	3482
6.717.2.2	~SessionIdMarshaller . . . . .	3482
6.717.3	Member Function Documentation . . . . .	3482
6.717.3.1	createObject . . . . .	3482
6.717.3.2	getDataStructureType . . . . .	3482
6.717.3.3	looseMarshal . . . . .	3482
6.717.3.4	looseUnmarshal . . . . .	3483
6.717.3.5	tightMarshal1 . . . . .	3483
6.717.3.6	tightMarshal2 . . . . .	3484
6.717.3.7	tightUnmarshal . . . . .	3484
6.718	activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller Class	
	Reference . . . . .	3485
6.718.1	Detailed Description . . . . .	3486
6.718.2	Constructor & Destructor Documentation . . . . .	3486
6.718.2.1	SessionIdMarshaller . . . . .	3486
6.718.2.2	~SessionIdMarshaller . . . . .	3486
6.718.3	Member Function Documentation . . . . .	3486
6.718.3.1	createObject . . . . .	3486
6.718.3.2	getDataStructureType . . . . .	3486
6.718.3.3	looseMarshal . . . . .	3486
6.718.3.4	looseUnmarshal . . . . .	3487
6.718.3.5	tightMarshal1 . . . . .	3487
6.718.3.6	tightMarshal2 . . . . .	3488
6.718.3.7	tightUnmarshal . . . . .	3488
6.719	activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller Class	
	Reference . . . . .	3489
6.719.1	Detailed Description . . . . .	3490
6.719.2	Constructor & Destructor Documentation . . . . .	3490
6.719.2.1	SessionIdMarshaller . . . . .	3490
6.719.2.2	~SessionIdMarshaller . . . . .	3490
6.719.3	Member Function Documentation . . . . .	3490
6.719.3.1	createObject . . . . .	3490
6.719.3.2	getDataStructureType . . . . .	3490
6.719.3.3	looseMarshal . . . . .	3490

6.719.3.4 looseUnmarshal . . . . .	3491
6.719.3.5 tightMarshal1 . . . . .	3491
6.719.3.6 tightMarshal2 . . . . .	3492
6.719.3.7 tightUnmarshal . . . . .	3492
6.720activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller Class	
Reference . . . . .	3493
6.720.1 Detailed Description . . . . .	3494
6.720.2 Constructor & Destructor Documentation . . . . .	3494
6.720.2.1 SessionIdMarshaller . . . . .	3494
6.720.2.2 ~SessionIdMarshaller . . . . .	3494
6.720.3 Member Function Documentation . . . . .	3494
6.720.3.1 createObject . . . . .	3494
6.720.3.2 getDataStructureType . . . . .	3494
6.720.3.3 looseMarshal . . . . .	3494
6.720.3.4 looseUnmarshal . . . . .	3495
6.720.3.5 tightMarshal1 . . . . .	3495
6.720.3.6 tightMarshal2 . . . . .	3496
6.720.3.7 tightUnmarshal . . . . .	3496
6.721activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller Class	
Reference . . . . .	3497
6.721.1 Detailed Description . . . . .	3498
6.721.2 Constructor & Destructor Documentation . . . . .	3498
6.721.2.1 SessionIdMarshaller . . . . .	3498
6.721.2.2 ~SessionIdMarshaller . . . . .	3498
6.721.3 Member Function Documentation . . . . .	3498
6.721.3.1 createObject . . . . .	3498
6.721.3.2 getDataStructureType . . . . .	3498
6.721.3.3 looseMarshal . . . . .	3498
6.721.3.4 looseUnmarshal . . . . .	3499
6.721.3.5 tightMarshal1 . . . . .	3499
6.721.3.6 tightMarshal2 . . . . .	3500
6.721.3.7 tightUnmarshal . . . . .	3500
6.722activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller Class	
Reference . . . . .	3501
6.722.1 Detailed Description . . . . .	3502
6.722.2 Constructor & Destructor Documentation . . . . .	3502
6.722.2.1 SessionIdMarshaller . . . . .	3502
6.722.2.2 ~SessionIdMarshaller . . . . .	3502
6.722.3 Member Function Documentation . . . . .	3502
6.722.3.1 createObject . . . . .	3502
6.722.3.2 getDataStructureType . . . . .	3502
6.722.3.3 looseMarshal . . . . .	3502
6.722.3.4 looseUnmarshal . . . . .	3503
6.722.3.5 tightMarshal1 . . . . .	3503
6.722.3.6 tightMarshal2 . . . . .	3504
6.722.3.7 tightUnmarshal . . . . .	3504
6.723activemq::commands::SessionInfo Class Reference . . . . .	3505
6.723.1 Constructor & Destructor Documentation . . . . .	3506
6.723.1.1 SessionInfo . . . . .	3506
6.723.1.2 ~SessionInfo . . . . .	3506

6.723.2 Member Function Documentation . . . . .	3506
6.723.2.1 cloneDataStructure . . . . .	3506
6.723.2.2 copyDataStructure . . . . .	3506
6.723.2.3 createRemoveCommand . . . . .	3507
6.723.2.4 equals . . . . .	3507
6.723.2.5 getAckMode . . . . .	3507
6.723.2.6 getDataStructureType . . . . .	3507
6.723.2.7 getSessionId . . . . .	3507
6.723.2.8 getSessionId . . . . .	3507
6.723.2.9 setAckMode . . . . .	3507
6.723.2.10setSessionId . . . . .	3507
6.723.2.11toString . . . . .	3507
6.723.2.12visit . . . . .	3508
6.723.3 Field Documentation . . . . .	3508
6.723.3.1 ID_SESSIONINFO . . . . .	3508
6.723.3.2 sessionId . . . . .	3508
6.724activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller Class	
Reference . . . . .	3508
6.724.1 Detailed Description . . . . .	3509
6.724.2 Constructor & Destructor Documentation . . . . .	3510
6.724.2.1 SessionInfoMarshaller . . . . .	3510
6.724.2.2 ~SessionInfoMarshaller . . . . .	3510
6.724.3 Member Function Documentation . . . . .	3510
6.724.3.1 createObject . . . . .	3510
6.724.3.2 getDataStructureType . . . . .	3510
6.724.3.3 looseMarshal . . . . .	3510
6.724.3.4 looseUnmarshal . . . . .	3511
6.724.3.5 tightMarshal1 . . . . .	3511
6.724.3.6 tightMarshal2 . . . . .	3512
6.724.3.7 tightUnmarshal . . . . .	3512
6.725activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller Class	
Reference . . . . .	3513
6.725.1 Detailed Description . . . . .	3514
6.725.2 Constructor & Destructor Documentation . . . . .	3514
6.725.2.1 SessionInfoMarshaller . . . . .	3514
6.725.2.2 ~SessionInfoMarshaller . . . . .	3514
6.725.3 Member Function Documentation . . . . .	3514
6.725.3.1 createObject . . . . .	3514
6.725.3.2 getDataStructureType . . . . .	3514
6.725.3.3 looseMarshal . . . . .	3514
6.725.3.4 looseUnmarshal . . . . .	3515
6.725.3.5 tightMarshal1 . . . . .	3515
6.725.3.6 tightMarshal2 . . . . .	3516
6.725.3.7 tightUnmarshal . . . . .	3516
6.726activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller Class	
Reference . . . . .	3517
6.726.1 Detailed Description . . . . .	3518
6.726.2 Constructor & Destructor Documentation . . . . .	3518
6.726.2.1 SessionInfoMarshaller . . . . .	3518
6.726.2.2 ~SessionInfoMarshaller . . . . .	3518



6.726.3 Member Function Documentation . . . . .	3518
6.726.3.1 createObject . . . . .	3518
6.726.3.2 getDataStructureType . . . . .	3518
6.726.3.3 looseMarshal . . . . .	3519
6.726.3.4 looseUnmarshal . . . . .	3519
6.726.3.5 tightMarshal1 . . . . .	3520
6.726.3.6 tightMarshal2 . . . . .	3520
6.726.3.7 tightUnmarshal . . . . .	3521
6.727activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller Class	
Reference . . . . .	3521
6.727.1 Detailed Description . . . . .	3522
6.727.2 Constructor & Destructor Documentation . . . . .	3522
6.727.2.1 SessionInfoMarshaller . . . . .	3522
6.727.2.2 ~SessionInfoMarshaller . . . . .	3522
6.727.3 Member Function Documentation . . . . .	3522
6.727.3.1 createObject . . . . .	3522
6.727.3.2 getDataStructureType . . . . .	3523
6.727.3.3 looseMarshal . . . . .	3523
6.727.3.4 looseUnmarshal . . . . .	3523
6.727.3.5 tightMarshal1 . . . . .	3524
6.727.3.6 tightMarshal2 . . . . .	3524
6.727.3.7 tightUnmarshal . . . . .	3525
6.728activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller Class	
Reference . . . . .	3525
6.728.1 Detailed Description . . . . .	3526
6.728.2 Constructor & Destructor Documentation . . . . .	3527
6.728.2.1 SessionInfoMarshaller . . . . .	3527
6.728.2.2 ~SessionInfoMarshaller . . . . .	3527
6.728.3 Member Function Documentation . . . . .	3527
6.728.3.1 createObject . . . . .	3527
6.728.3.2 getDataStructureType . . . . .	3527
6.728.3.3 looseMarshal . . . . .	3527
6.728.3.4 looseUnmarshal . . . . .	3528
6.728.3.5 tightMarshal1 . . . . .	3528
6.728.3.6 tightMarshal2 . . . . .	3529
6.728.3.7 tightUnmarshal . . . . .	3529
6.729activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller Class	
Reference . . . . .	3530
6.729.1 Detailed Description . . . . .	3531
6.729.2 Constructor & Destructor Documentation . . . . .	3531
6.729.2.1 SessionInfoMarshaller . . . . .	3531
6.729.2.2 ~SessionInfoMarshaller . . . . .	3531
6.729.3 Member Function Documentation . . . . .	3531
6.729.3.1 createObject . . . . .	3531
6.729.3.2 getDataStructureType . . . . .	3531
6.729.3.3 looseMarshal . . . . .	3531
6.729.3.4 looseUnmarshal . . . . .	3532
6.729.3.5 tightMarshal1 . . . . .	3532
6.729.3.6 tightMarshal2 . . . . .	3533
6.729.3.7 tightUnmarshal . . . . .	3533

6.730	activemq::cmsutil::SessionPool Class Reference . . . . .	3534
6.730.1	Detailed Description . . . . .	3534
6.730.2	Constructor & Destructor Documentation . . . . .	3535
6.730.2.1	SessionPool . . . . .	3535
6.730.2.2	SessionPool . . . . .	3535
6.730.2.3	~SessionPool . . . . .	3535
6.730.3	Member Function Documentation . . . . .	3535
6.730.3.1	getResourceLifecycleManager . . . . .	3535
6.730.3.2	operator= . . . . .	3535
6.730.3.3	returnSession . . . . .	3535
6.730.3.4	takeSession . . . . .	3536
6.731	activemq::state::SessionState Class Reference . . . . .	3536
6.731.1	Constructor & Destructor Documentation . . . . .	3537
6.731.1.1	SessionState . . . . .	3537
6.731.1.2	~SessionState . . . . .	3537
6.731.2	Member Function Documentation . . . . .	3537
6.731.2.1	addConsumer . . . . .	3537
6.731.2.2	addProducer . . . . .	3537
6.731.2.3	checkShutdown . . . . .	3537
6.731.2.4	getConsumerState . . . . .	3537
6.731.2.5	getConsumerStates . . . . .	3537
6.731.2.6	getInfo . . . . .	3537
6.731.2.7	getProducerState . . . . .	3537
6.731.2.8	getProducerStates . . . . .	3537
6.731.2.9	removeConsumer . . . . .	3537
6.731.2.10	removeProducer . . . . .	3537
6.731.2.11	shutdown . . . . .	3537
6.731.2.12	toString . . . . .	3537
6.732	decaf::util::Set< E > Class Template Reference . . . . .	3538
6.732.1	Detailed Description . . . . .	3538
6.732.2	Constructor & Destructor Documentation . . . . .	3538
6.732.2.1	~Set . . . . .	3538
6.733	decaf::lang::Short Class Reference . . . . .	3538
6.733.1	Constructor & Destructor Documentation . . . . .	3541
6.733.1.1	Short . . . . .	3541
6.733.1.2	Short . . . . .	3541
6.733.1.3	~Short . . . . .	3541
6.733.2	Member Function Documentation . . . . .	3541
6.733.2.1	byteValue . . . . .	3541
6.733.2.2	compareTo . . . . .	3541
6.733.2.3	compareTo . . . . .	3542
6.733.2.4	decode . . . . .	3542
6.733.2.5	doubleValue . . . . .	3542
6.733.2.6	equals . . . . .	3543
6.733.2.7	equals . . . . .	3543
6.733.2.8	floatValue . . . . .	3543
6.733.2.9	intValue . . . . .	3543
6.733.2.10	longValue . . . . .	3543
6.733.2.11	operator< . . . . .	3544
6.733.2.12	operator< . . . . .	3544

6.733.2.13operator== . . . . .	3544
6.733.2.14operator== . . . . .	3545
6.733.2.15parseShort . . . . .	3545
6.733.2.16parseShort . . . . .	3546
6.733.2.17reverseBytes . . . . .	3546
6.733.2.18shortValue . . . . .	3546
6.733.2.19toString . . . . .	3546
6.733.2.20toString . . . . .	3547
6.733.2.21valueOf . . . . .	3547
6.733.2.22valueOf . . . . .	3547
6.733.2.23valueOf . . . . .	3547
6.733.3 Field Documentation . . . . .	3548
6.733.3.1 MAX_VALUE . . . . .	3548
6.733.3.2 MIN_VALUE . . . . .	3548
6.733.3.3 SIZE . . . . .	3548
6.734decaf::internal::nio::ShortArrayBuffer Class Reference . . . . .	3548
6.734.1 Constructor & Destructor Documentation . . . . .	3553
6.734.1.1 ShortArrayBuffer . . . . .	3553
6.734.1.2 ShortArrayBuffer . . . . .	3553
6.734.1.3 ShortArrayBuffer . . . . .	3554
6.734.1.4 ShortArrayBuffer . . . . .	3554
6.734.1.5 ~ShortArrayBuffer . . . . .	3554
6.734.2 Member Function Documentation . . . . .	3554
6.734.2.1 array . . . . .	3554
6.734.2.2 arrayOffset . . . . .	3555
6.734.2.3 asReadOnlyBuffer . . . . .	3555
6.734.2.4 compact . . . . .	3556
6.734.2.5 duplicate . . . . .	3556
6.734.2.6 get . . . . .	3557
6.734.2.7 get . . . . .	3557
6.734.2.8 hasArray . . . . .	3558
6.734.2.9 isReadOnly . . . . .	3558
6.734.2.10put . . . . .	3558
6.734.2.11put . . . . .	3559
6.734.2.12setReadOnly . . . . .	3559
6.734.2.13slice . . . . .	3559
6.735decaf::nio::ShortBuffer Class Reference . . . . .	3560
6.735.1 Detailed Description . . . . .	3562
6.735.2 Constructor & Destructor Documentation . . . . .	3563
6.735.2.1 ShortBuffer . . . . .	3563
6.735.2.2 ~ShortBuffer . . . . .	3563
6.735.3 Member Function Documentation . . . . .	3563
6.735.3.1 allocate . . . . .	3563
6.735.3.2 array . . . . .	3563
6.735.3.3 arrayOffset . . . . .	3564
6.735.3.4 asReadOnlyBuffer . . . . .	3564
6.735.3.5 compact . . . . .	3565
6.735.3.6 compareTo . . . . .	3565
6.735.3.7 duplicate . . . . .	3565
6.735.3.8 equals . . . . .	3566

6.735.3.9	get	3566
6.735.3.10	get	3566
6.735.3.11	get	3567
6.735.3.12	get	3567
6.735.3.13	hasArray	3568
6.735.3.14	operator<	3568
6.735.3.15	operator==	3568
6.735.3.16	put	3568
6.735.3.17	put	3569
6.735.3.18	put	3569
6.735.3.19	put	3570
6.735.3.20	put	3570
6.735.3.21	slice	3571
6.735.3.22	toString	3571
6.735.3.23	wrap	3572
6.735.3.24	wrap	3572
6.736	activemq::commands::ShutdownInfo Class Reference	3573
6.736.1	Constructor & Destructor Documentation	3574
6.736.1.1	ShutdownInfo	3574
6.736.1.2	~ShutdownInfo	3574
6.736.2	Member Function Documentation	3574
6.736.2.1	cloneDataStructure	3574
6.736.2.2	copyDataStructure	3574
6.736.2.3	equals	3574
6.736.2.4	getDataStructureType	3575
6.736.2.5	isShutdownInfo	3575
6.736.2.6	toString	3575
6.736.2.7	visit	3575
6.736.3	Field Documentation	3576
6.736.3.1	ID_SHUTDOWNINFO	3576
6.737	activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller	
	Class Reference	3576
6.737.1	Detailed Description	3577
6.737.2	Constructor & Destructor Documentation	3577
6.737.2.1	ShutdownInfoMarshaller	3577
6.737.2.2	~ShutdownInfoMarshaller	3577
6.737.3	Member Function Documentation	3577
6.737.3.1	createObject	3577
6.737.3.2	getDataStructureType	3577
6.737.3.3	looseMarshal	3578
6.737.3.4	looseUnmarshal	3578
6.737.3.5	tightMarshal1	3579
6.737.3.6	tightMarshal2	3579
6.737.3.7	tightUnmarshal	3580
6.738	activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller	
	Class Reference	3580
6.738.1	Detailed Description	3581
6.738.2	Constructor & Destructor Documentation	3581
6.738.2.1	ShutdownInfoMarshaller	3581
6.738.2.2	~ShutdownInfoMarshaller	3581

6.738.3 Member Function Documentation . . . . .	3581
6.738.3.1 createObject . . . . .	3581
6.738.3.2 getDataStructureType . . . . .	3582
6.738.3.3 looseMarshal . . . . .	3582
6.738.3.4 looseUnmarshal . . . . .	3582
6.738.3.5 tightMarshal1 . . . . .	3583
6.738.3.6 tightMarshal2 . . . . .	3583
6.738.3.7 tightUnmarshal . . . . .	3584
6.739activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller	
Class Reference . . . . .	3584
6.739.1 Detailed Description . . . . .	3585
6.739.2 Constructor & Destructor Documentation . . . . .	3586
6.739.2.1 ShutdownInfoMarshaller . . . . .	3586
6.739.2.2 ~ShutdownInfoMarshaller . . . . .	3586
6.739.3 Member Function Documentation . . . . .	3586
6.739.3.1 createObject . . . . .	3586
6.739.3.2 getDataStructureType . . . . .	3586
6.739.3.3 looseMarshal . . . . .	3586
6.739.3.4 looseUnmarshal . . . . .	3587
6.739.3.5 tightMarshal1 . . . . .	3587
6.739.3.6 tightMarshal2 . . . . .	3588
6.739.3.7 tightUnmarshal . . . . .	3588
6.740activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller	
Class Reference . . . . .	3589
6.740.1 Detailed Description . . . . .	3590
6.740.2 Constructor & Destructor Documentation . . . . .	3590
6.740.2.1 ShutdownInfoMarshaller . . . . .	3590
6.740.2.2 ~ShutdownInfoMarshaller . . . . .	3590
6.740.3 Member Function Documentation . . . . .	3590
6.740.3.1 createObject . . . . .	3590
6.740.3.2 getDataStructureType . . . . .	3590
6.740.3.3 looseMarshal . . . . .	3590
6.740.3.4 looseUnmarshal . . . . .	3591
6.740.3.5 tightMarshal1 . . . . .	3591
6.740.3.6 tightMarshal2 . . . . .	3592
6.740.3.7 tightUnmarshal . . . . .	3592
6.741activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller	
Class Reference . . . . .	3593
6.741.1 Detailed Description . . . . .	3594
6.741.2 Constructor & Destructor Documentation . . . . .	3594
6.741.2.1 ShutdownInfoMarshaller . . . . .	3594
6.741.2.2 ~ShutdownInfoMarshaller . . . . .	3594
6.741.3 Member Function Documentation . . . . .	3594
6.741.3.1 createObject . . . . .	3594
6.741.3.2 getDataStructureType . . . . .	3594
6.741.3.3 looseMarshal . . . . .	3595
6.741.3.4 looseUnmarshal . . . . .	3595
6.741.3.5 tightMarshal1 . . . . .	3596
6.741.3.6 tightMarshal2 . . . . .	3596
6.741.3.7 tightUnmarshal . . . . .	3597

6.742activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	
Class Reference . . . . .	3597
6.742.1 Detailed Description . . . . .	3598
6.742.2 Constructor & Destructor Documentation . . . . .	3598
6.742.2.1 ShutdownInfoMarshaller . . . . .	3598
6.742.2.2 ~ShutdownInfoMarshaller . . . . .	3598
6.742.3 Member Function Documentation . . . . .	3598
6.742.3.1 createObject . . . . .	3598
6.742.3.2 getDataStructureType . . . . .	3599
6.742.3.3 looseMarshal . . . . .	3599
6.742.3.4 looseUnmarshal . . . . .	3599
6.742.3.5 tightMarshal1 . . . . .	3600
6.742.3.6 tightMarshal2 . . . . .	3600
6.742.3.7 tightUnmarshal . . . . .	3601
6.743decaf::security::SignatureException Class Reference . . . . .	3601
6.743.1 Constructor & Destructor Documentation . . . . .	3602
6.743.1.1 SignatureException . . . . .	3602
6.743.1.2 SignatureException . . . . .	3602
6.743.1.3 SignatureException . . . . .	3602
6.743.1.4 SignatureException . . . . .	3603
6.743.1.5 SignatureException . . . . .	3603
6.743.1.6 SignatureException . . . . .	3603
6.743.1.7 ~SignatureException . . . . .	3604
6.743.2 Member Function Documentation . . . . .	3604
6.743.2.1 clone . . . . .	3604
6.744decaf::util::logging::SimpleFormatter Class Reference . . . . .	3604
6.744.1 Detailed Description . . . . .	3604
6.744.2 Constructor & Destructor Documentation . . . . .	3605
6.744.2.1 SimpleFormatter . . . . .	3605
6.744.2.2 ~SimpleFormatter . . . . .	3605
6.744.3 Member Function Documentation . . . . .	3605
6.744.3.1 format . . . . .	3605
6.745decaf::util::logging::SimpleLogger Class Reference . . . . .	3605
6.745.1 Constructor & Destructor Documentation . . . . .	3606
6.745.1.1 SimpleLogger . . . . .	3606
6.745.1.2 ~SimpleLogger . . . . .	3606
6.745.2 Member Function Documentation . . . . .	3606
6.745.2.1 debug . . . . .	3606
6.745.2.2 error . . . . .	3606
6.745.2.3 fatal . . . . .	3606
6.745.2.4 info . . . . .	3607
6.745.2.5 log . . . . .	3607
6.745.2.6 mark . . . . .	3607
6.745.2.7 warn . . . . .	3607
6.746decaf::net::Socket Class Reference . . . . .	3607
6.746.1 Detailed Description . . . . .	3611
6.746.2 Constructor & Destructor Documentation . . . . .	3611
6.746.2.1 Socket . . . . .	3611
6.746.2.2 Socket . . . . .	3611
6.746.2.3 Socket . . . . .	3612

6.746.2.4 Socket . . . . .	3612
6.746.2.5 Socket . . . . .	3613
6.746.2.6 Socket . . . . .	3613
6.746.2.7 ~Socket . . . . .	3614
6.746.3 Member Function Documentation . . . . .	3614
6.746.3.1 accepted . . . . .	3614
6.746.3.2 bind . . . . .	3614
6.746.3.3 checkClosed . . . . .	3614
6.746.3.4 close . . . . .	3614
6.746.3.5 connect . . . . .	3615
6.746.3.6 connect . . . . .	3615
6.746.3.7 ensureCreated . . . . .	3616
6.746.3.8 getInetAddress . . . . .	3616
6.746.3.9 getInputStream . . . . .	3616
6.746.3.10 getKeepAlive . . . . .	3616
6.746.3.11 getLocalAddress . . . . .	3617
6.746.3.12 getLocalPort . . . . .	3617
6.746.3.13 getOOBInline . . . . .	3617
6.746.3.14 getOutputStream . . . . .	3617
6.746.3.15 getPort . . . . .	3618
6.746.3.16 getReceiveBufferSize . . . . .	3618
6.746.3.17 getReuseAddress . . . . .	3618
6.746.3.18 getSendBufferSize . . . . .	3618
6.746.3.19 getSoLinger . . . . .	3619
6.746.3.20 getSoTimeout . . . . .	3619
6.746.3.21 getTcpNoDelay . . . . .	3619
6.746.3.22 getTrafficClass . . . . .	3620
6.746.3.23 initSocketImpl . . . . .	3620
6.746.3.24 isBound . . . . .	3620
6.746.3.25 isClosed . . . . .	3620
6.746.3.26 isConnected . . . . .	3620
6.746.3.27 isInputShutdown . . . . .	3621
6.746.3.28 isOutputShutdown . . . . .	3621
6.746.3.29 sendUrgentData . . . . .	3621
6.746.3.30 setKeepAlive . . . . .	3621
6.746.3.31 setOOBInline . . . . .	3621
6.746.3.32 setReceiveBufferSize . . . . .	3622
6.746.3.33 setReuseAddress . . . . .	3622
6.746.3.34 setSendBufferSize . . . . .	3622
6.746.3.35 setSocketImplFactory . . . . .	3623
6.746.3.36 setSoLinger . . . . .	3623
6.746.3.37 setSoTimeout . . . . .	3624
6.746.3.38 setTcpNoDelay . . . . .	3624
6.746.3.39 setTrafficClass . . . . .	3624
6.746.3.40 shutdownInput . . . . .	3625
6.746.3.41 shutdownOutput . . . . .	3625
6.746.3.42 toString . . . . .	3625
6.746.4 Friends And Related Function Documentation . . . . .	3625
6.746.4.1 ServerSocket . . . . .	3625
6.746.5 Field Documentation . . . . .	3625

6.746.5.1 impl . . . . .	3625
6.747decaf::net::SocketAddress Class Reference . . . . .	3626
6.747.1 Detailed Description . . . . .	3626
6.747.2 Constructor & Destructor Documentation . . . . .	3626
6.747.2.1 ~SocketAddress . . . . .	3626
6.748decaf::net::SocketError Class Reference . . . . .	3626
6.748.1 Detailed Description . . . . .	3627
6.748.2 Member Function Documentation . . . . .	3627
6.748.2.1 getErrorCode . . . . .	3627
6.748.2.2 getErrorString . . . . .	3627
6.749decaf::net::SocketException Class Reference . . . . .	3627
6.749.1 Detailed Description . . . . .	3628
6.749.2 Constructor & Destructor Documentation . . . . .	3628
6.749.2.1 SocketException . . . . .	3628
6.749.2.2 SocketException . . . . .	3628
6.749.2.3 SocketException . . . . .	3628
6.749.2.4 SocketException . . . . .	3628
6.749.2.5 SocketException . . . . .	3628
6.749.2.6 SocketException . . . . .	3629
6.749.2.7 ~SocketException . . . . .	3629
6.749.3 Member Function Documentation . . . . .	3629
6.749.3.1 clone . . . . .	3629
6.750decaf::net::SocketFactory Class Reference . . . . .	3629
6.750.1 Detailed Description . . . . .	3630
6.750.2 Constructor & Destructor Documentation . . . . .	3631
6.750.2.1 SocketFactory . . . . .	3631
6.750.2.2 ~SocketFactory . . . . .	3631
6.750.3 Member Function Documentation . . . . .	3631
6.750.3.1 createSocket . . . . .	3631
6.750.3.2 createSocket . . . . .	3631
6.750.3.3 createSocket . . . . .	3632
6.750.3.4 createSocket . . . . .	3632
6.750.3.5 createSocket . . . . .	3633
6.750.3.6 getDefault . . . . .	3633
6.751decaf::internal::net::SocketFileDescriptor Class Reference . . . . .	3634
6.751.1 Detailed Description . . . . .	3634
6.751.2 Constructor & Destructor Documentation . . . . .	3635
6.751.2.1 SocketFileDescriptor . . . . .	3635
6.751.2.2 ~SocketFileDescriptor . . . . .	3635
6.751.3 Member Function Documentation . . . . .	3635
6.751.3.1 getValue . . . . .	3635
6.752decaf::net::SocketImpl Class Reference . . . . .	3635
6.752.1 Detailed Description . . . . .	3637
6.752.2 Constructor & Destructor Documentation . . . . .	3638
6.752.2.1 SocketImpl . . . . .	3638
6.752.2.2 ~SocketImpl . . . . .	3638
6.752.3 Member Function Documentation . . . . .	3638
6.752.3.1 accept . . . . .	3638
6.752.3.2 available . . . . .	3638
6.752.3.3 bind . . . . .	3638



6.752.3.4	close	3639
6.752.3.5	connect	3639
6.752.3.6	create	3640
6.752.3.7	getFileDescriptor	3640
6.752.3.8	getInetAddress	3640
6.752.3.9	getInputStream	3640
6.752.3.10	getLocalAddress	3641
6.752.3.11	getLocalPort	3641
6.752.3.12	getOption	3641
6.752.3.13	getOutputStream	3641
6.752.3.14	getPort	3642
6.752.3.15	listen	3642
6.752.3.16	sendUrgentData	3642
6.752.3.17	setOption	3643
6.752.3.18	shutdownInput	3643
6.752.3.19	shutdownOutput	3643
6.752.3.20	supportsUrgentData	3644
6.752.3.21	toString	3644
6.752.4	Field Documentation	3644
6.752.4.1	address	3644
6.752.4.2	fd	3644
6.752.4.3	localPort	3644
6.752.4.4	port	3644
6.753	decaf::net::SocketImplFactory Class Reference	3644
6.753.1	Detailed Description	3645
6.753.2	Constructor & Destructor Documentation	3645
6.753.2.1	~SocketImplFactory	3645
6.753.3	Member Function Documentation	3645
6.753.3.1	createSocketImpl	3645
6.754	decaf::net::SocketOptions Class Reference	3645
6.754.1	Detailed Description	3647
6.754.2	Constructor & Destructor Documentation	3647
6.754.2.1	~SocketOptions	3647
6.754.3	Field Documentation	3647
6.754.3.1	SOCKET_OPTION_BINDADDR	3647
6.754.3.2	SOCKET_OPTION_BROADCAST	3647
6.754.3.3	SOCKET_OPTION_IP_MULTICAST_IF	3648
6.754.3.4	SOCKET_OPTION_IP_MULTICAST_IF2	3648
6.754.3.5	SOCKET_OPTION_IP_MULTICAST_LOOP	3648
6.754.3.6	SOCKET_OPTION_IP_TOS	3648
6.754.3.7	SOCKET_OPTION_KEEPALIVE	3648
6.754.3.8	SOCKET_OPTION_LINGER	3649
6.754.3.9	SOCKET_OPTION_OOINLINE	3649
6.754.3.10	SOCKET_OPTION_RCVBUF	3649
6.754.3.11	SOCKET_OPTION_REUSEADDR	3649
6.754.3.12	SOCKET_OPTION_SNDBUF	3649
6.754.3.13	SOCKET_OPTION_TCP_NODELAY	3650
6.754.3.14	SOCKET_OPTION_TIMEOUT	3650
6.755	decaf::net::SocketTimeoutException Class Reference	3650
6.755.1	Constructor & Destructor Documentation	3651

6.755.1.1	SocketTimeoutException	3651
6.755.1.2	SocketTimeoutException	3651
6.755.1.3	SocketTimeoutException	3651
6.755.1.4	SocketTimeoutException	3651
6.755.1.5	SocketTimeoutException	3652
6.755.1.6	SocketTimeoutException	3652
6.755.1.7	~SocketTimeoutException	3652
6.755.2	Member Function Documentation	3652
6.755.2.1	clone	3652
6.756	decaf::net::ssl::SSLContext Class Reference	3653
6.756.1	Detailed Description	3653
6.756.2	Constructor & Destructor Documentation	3654
6.756.2.1	SSLContext	3654
6.756.2.2	~SSLContext	3654
6.756.3	Member Function Documentation	3654
6.756.3.1	getDefault	3654
6.756.3.2	getDefaultSSLParameters	3654
6.756.3.3	getServerSocketFactory	3654
6.756.3.4	getSocketFactory	3655
6.756.3.5	getSupportedSSLParameters	3655
6.756.3.6	setDefault	3655
6.757	decaf::net::ssl::SSLContextSpi Class Reference	3655
6.757.1	Detailed Description	3656
6.757.2	Constructor & Destructor Documentation	3656
6.757.2.1	~SSLContextSpi	3656
6.757.3	Member Function Documentation	3656
6.757.3.1	providerGetDefaultSSLParameters	3656
6.757.3.2	providerGetServerSocketFactory	3657
6.757.3.3	providerGetSocketFactory	3657
6.757.3.4	providerGetSupportedSSLParameters	3658
6.757.3.5	providerInit	3658
6.758	decaf::net::ssl::SSLParameters Class Reference	3658
6.758.1	Constructor & Destructor Documentation	3659
6.758.1.1	SSLParameters	3659
6.758.1.2	SSLParameters	3659
6.758.1.3	SSLParameters	3660
6.758.1.4	~SSLParameters	3660
6.758.2	Member Function Documentation	3660
6.758.2.1	getCipherSuites	3660
6.758.2.2	getNeedClientAuth	3660
6.758.2.3	getProtocols	3660
6.758.2.4	getWantClientAuth	3660
6.758.2.5	setCipherSuites	3661
6.758.2.6	setNeedClientAuth	3661
6.758.2.7	setProtocols	3661
6.758.2.8	setWantClientAuth	3661
6.759	decaf::net::ssl::SSLServerSocket Class Reference	3662
6.759.1	Detailed Description	3663
6.759.2	Constructor & Destructor Documentation	3663
6.759.2.1	SSLServerSocket	3663

6.759.2.2	SSLServerSocket	3663
6.759.2.3	SSLServerSocket	3664
6.759.2.4	SSLServerSocket	3664
6.759.2.5	~SSLServerSocket	3665
6.759.3	Member Function Documentation	3665
6.759.3.1	getEnabledCipherSuites	3665
6.759.3.2	getEnabledProtocols	3665
6.759.3.3	getNeedClientAuth	3665
6.759.3.4	getSupportedCipherSuites	3666
6.759.3.5	getSupportedProtocols	3666
6.759.3.6	getWantClientAuth	3666
6.759.3.7	setEnabledCipherSuites	3666
6.759.3.8	setEnabledProtocols	3667
6.759.3.9	setNeedClientAuth	3667
6.759.3.10	setWantClientAuth	3667
6.760	decaf::net::ssl::SSLServerSocketFactory Class Reference	3668
6.760.1	Detailed Description	3668
6.760.2	Constructor & Destructor Documentation	3669
6.760.2.1	SSLServerSocketFactory	3669
6.760.2.2	~SSLServerSocketFactory	3669
6.760.3	Member Function Documentation	3669
6.760.3.1	getDefault	3669
6.760.3.2	getDefaultCipherSuites	3669
6.760.3.3	getSupportedCipherSuites	3670
6.761	decaf::net::ssl::SSLSocket Class Reference	3670
6.761.1	Detailed Description	3672
6.761.2	Constructor & Destructor Documentation	3672
6.761.2.1	SSLSocket	3672
6.761.2.2	SSLSocket	3672
6.761.2.3	SSLSocket	3673
6.761.2.4	SSLSocket	3673
6.761.2.5	SSLSocket	3674
6.761.2.6	~SSLSocket	3674
6.761.3	Member Function Documentation	3674
6.761.3.1	getEnabledCipherSuites	3674
6.761.3.2	getEnabledProtocols	3674
6.761.3.3	getNeedClientAuth	3675
6.761.3.4	getSSLParameters	3675
6.761.3.5	getSupportedCipherSuites	3675
6.761.3.6	getSupportedProtocols	3675
6.761.3.7	getUseClientMode	3676
6.761.3.8	getWantClientAuth	3676
6.761.3.9	setEnabledCipherSuites	3676
6.761.3.10	setEnabledProtocols	3677
6.761.3.11	setNeedClientAuth	3677
6.761.3.12	setSSLParameters	3677
6.761.3.13	setUseClientMode	3678
6.761.3.14	setWantClientAuth	3678
6.761.3.15	startHandshake	3679
6.762	decaf::net::ssl::SSLSocketFactory Class Reference	3679

6.762.1 Detailed Description . . . . .	3680
6.762.2 Constructor & Destructor Documentation . . . . .	3680
6.762.2.1 SSLSocketFactory . . . . .	3680
6.762.2.2 ~SSLSocketFactory . . . . .	3680
6.762.3 Member Function Documentation . . . . .	3680
6.762.3.1 createSocket . . . . .	3680
6.762.3.2 getDefault . . . . .	3681
6.762.3.3 getDefaultCipherSuites . . . . .	3681
6.762.3.4 getSupportedCipherSuites . . . . .	3682
6.763activemq::transport::tcp::SslTransport Class Reference . . . . .	3682
6.763.1 Detailed Description . . . . .	3683
6.763.2 Constructor & Destructor Documentation . . . . .	3683
6.763.2.1 SslTransport . . . . .	3683
6.763.2.2 ~SslTransport . . . . .	3683
6.763.3 Member Function Documentation . . . . .	3683
6.763.3.1 configureSocket . . . . .	3683
6.763.3.2 createSocket . . . . .	3683
6.764activemq::transport::tcp::SslTransportFactory Class Reference . . . . .	3684
6.764.1 Constructor & Destructor Documentation . . . . .	3685
6.764.1.1 ~SslTransportFactory . . . . .	3685
6.764.2 Member Function Documentation . . . . .	3685
6.764.2.1 doCreateComposite . . . . .	3685
6.765activemq::commands::BrokerError::StackTraceElement Struct Reference	3685
6.765.1 Field Documentation . . . . .	3686
6.765.1.1 className . . . . .	3686
6.765.1.2 fileName . . . . .	3686
6.765.1.3 lineNumber . . . . .	3686
6.765.1.4 methodName . . . . .	3686
6.766decaf::internal::io::StandardErrorOutputStream Class Reference . . . . .	3686
6.766.1 Detailed Description . . . . .	3687
6.766.2 Constructor & Destructor Documentation . . . . .	3687
6.766.2.1 StandardErrorOutputStream . . . . .	3687
6.766.2.2 ~StandardErrorOutputStream . . . . .	3687
6.766.3 Member Function Documentation . . . . .	3687
6.766.3.1 close . . . . .	3687
6.766.3.2 doWriteArrayBounded . . . . .	3687
6.766.3.3 doWriteByte . . . . .	3688
6.766.3.4 flush . . . . .	3688
6.767decaf::internal::io::StandardInputStream Class Reference . . . . .	3688
6.767.1 Constructor & Destructor Documentation . . . . .	3689
6.767.1.1 StandardInputStream . . . . .	3689
6.767.1.2 ~StandardInputStream . . . . .	3689
6.767.2 Member Function Documentation . . . . .	3689
6.767.2.1 available . . . . .	3689
6.767.2.2 doReadByte . . . . .	3689
6.768decaf::internal::io::StandardOutputStream Class Reference . . . . .	3689
6.768.1 Constructor & Destructor Documentation . . . . .	3690
6.768.1.1 StandardOutputStream . . . . .	3690
6.768.1.2 ~StandardOutputStream . . . . .	3690
6.768.2 Member Function Documentation . . . . .	3690

6.768.2.1	close	3690
6.768.2.2	doWriteArrayBounded	3691
6.768.2.3	doWriteByte	3691
6.768.2.4	flush	3691
6.769	cms::Startable Class Reference	3691
6.769.1	Detailed Description	3692
6.769.2	Constructor & Destructor Documentation	3692
6.769.2.1	~Startable	3692
6.769.3	Member Function Documentation	3692
6.769.3.1	start	3692
6.770	decaf::lang::STATIC_CAST_TOKEN Struct Reference	3692
6.771	activemq::core::ActiveMQConstants::StaticInitializer Class Reference	3693
6.771.1	Constructor & Destructor Documentation	3693
6.771.1.1	StaticInitializer	3693
6.771.1.2	~StaticInitializer	3693
6.771.2	Field Documentation	3693
6.771.2.1	destOptionMap	3693
6.771.2.2	destOptions	3693
6.771.2.3	uriParams	3693
6.771.2.4	uriParamsMap	3693
6.772	decaf::util::StlList< E > Class Template Reference	3694
6.772.1	Detailed Description	3699
6.772.2	Constructor & Destructor Documentation	3699
6.772.2.1	StlList	3699
6.772.2.2	StlList	3699
6.772.2.3	StlList	3700
6.772.2.4	~StlList	3700
6.772.3	Member Function Documentation	3700
6.772.3.1	add	3700
6.772.3.2	add	3701
6.772.3.3	addAll	3701
6.772.3.4	clear	3702
6.772.3.5	contains	3702
6.772.3.6	copy	3703
6.772.3.7	equals	3703
6.772.3.8	get	3703
6.772.3.9	indexOf	3703
6.772.3.10	isEmpty	3704
6.772.3.11	iterator	3704
6.772.3.12	iterator	3704
6.772.3.13	lastIndexOf	3704
6.772.3.14	listIterator	3705
6.772.3.15	listIterator	3705
6.772.3.16	listIterator	3706
6.772.3.17	listIterator	3706
6.772.3.18	remove	3706
6.772.3.19	remove	3706
6.772.3.20	set	3707
6.772.3.21	size	3708
6.773	decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference	3708

6.773.1 Detailed Description . . . . .	3712
6.773.2 Constructor & Destructor Documentation . . . . .	3713
6.773.2.1 StlMap . . . . .	3713
6.773.2.2 StlMap . . . . .	3713
6.773.2.3 StlMap . . . . .	3713
6.773.2.4 ~StlMap . . . . .	3713
6.773.3 Member Function Documentation . . . . .	3713
6.773.3.1 clear . . . . .	3713
6.773.3.2 containsKey . . . . .	3714
6.773.3.3 containsValue . . . . .	3714
6.773.3.4 copy . . . . .	3714
6.773.3.5 copy . . . . .	3714
6.773.3.6 equals . . . . .	3715
6.773.3.7 equals . . . . .	3715
6.773.3.8 get . . . . .	3715
6.773.3.9 get . . . . .	3716
6.773.3.10 isEmpty . . . . .	3716
6.773.3.11 keySet . . . . .	3716
6.773.3.12 lock . . . . .	3717
6.773.3.13 notify . . . . .	3717
6.773.3.14 notifyAll . . . . .	3717
6.773.3.15 put . . . . .	3718
6.773.3.16 putAll . . . . .	3718
6.773.3.17 putAll . . . . .	3718
6.773.3.18 remove . . . . .	3719
6.773.3.19 size . . . . .	3719
6.773.3.20 tryLock . . . . .	3719
6.773.3.21 unlock . . . . .	3720
6.773.3.22 values . . . . .	3720
6.773.3.23 wait . . . . .	3720
6.773.3.24 wait . . . . .	3721
6.773.3.25 wait . . . . .	3721
6.774 decaf::util::StlQueue< T > Class Template Reference . . . . .	3722
6.774.1 Detailed Description . . . . .	3724
6.774.2 Constructor & Destructor Documentation . . . . .	3725
6.774.2.1 StlQueue . . . . .	3725
6.774.2.2 ~StlQueue . . . . .	3725
6.774.3 Member Function Documentation . . . . .	3725
6.774.3.1 back . . . . .	3725
6.774.3.2 back . . . . .	3725
6.774.3.3 clear . . . . .	3725
6.774.3.4 empty . . . . .	3725
6.774.3.5 enqueueFront . . . . .	3725
6.774.3.6 front . . . . .	3726
6.774.3.7 front . . . . .	3726
6.774.3.8 getSafeValue . . . . .	3726
6.774.3.9 iterator . . . . .	3726
6.774.3.10 lock . . . . .	3727
6.774.3.11 notify . . . . .	3727
6.774.3.12 notifyAll . . . . .	3727

6.774.3.13	pop	3728
6.774.3.14	push	3728
6.774.3.15	reverse	3728
6.774.3.16	size	3728
6.774.3.17	toArray	3728
6.774.3.18	tryLock	3729
6.774.3.19	unlock	3729
6.774.3.20	wait	3729
6.774.3.21	wait	3730
6.774.3.22	wait	3730
6.775	decaf::util::StlSet< E > Class Template Reference	3731
6.775.1	Detailed Description	3733
6.775.2	Constructor & Destructor Documentation	3734
6.775.2.1	StlSet	3734
6.775.2.2	StlSet	3734
6.775.2.3	StlSet	3734
6.775.2.4	~StlSet	3734
6.775.3	Member Function Documentation	3734
6.775.3.1	add	3734
6.775.3.2	clear	3735
6.775.3.3	contains	3736
6.775.3.4	copy	3736
6.775.3.5	equals	3736
6.775.3.6	isEmpty	3736
6.775.3.7	iterator	3736
6.775.3.8	iterator	3736
6.775.3.9	remove	3737
6.775.3.10	size	3737
6.776	activemq::wireformat::stomp::StompCommandConstants Class Reference	3738
6.776.1	Field Documentation	3740
6.776.1.1	ABORT	3740
6.776.1.2	ACK	3740
6.776.1.3	ACK_AUTO	3740
6.776.1.4	ACK_CLIENT	3740
6.776.1.5	ACK_INDIVIDUAL	3740
6.776.1.6	BEGIN	3740
6.776.1.7	BYTES	3740
6.776.1.8	COMMIT	3740
6.776.1.9	CONNECT	3740
6.776.1.10	CONNECTED	3740
6.776.1.11	DISCONNECT	3740
6.776.1.12	ERROR_CMD	3740
6.776.1.13	HEADER_ACK	3740
6.776.1.14	HEADER_CLIENT_ID	3740
6.776.1.15	HEADER_CONSUMERPRIORITY	3740
6.776.1.16	HEADER_CONTENTLENGTH	3740
6.776.1.17	HEADER_CORRELATIONID	3740
6.776.1.18	HEADER_DESTINATION	3740
6.776.1.19	HEADER_DISPATCH_ASYNC	3740

6.776.1.20	HEADER_EXCLUSIVE	3740
6.776.1.21	HEADER_EXPIRES	3740
6.776.1.22	HEADER_ID	3740
6.776.1.23	HEADER_JMSPRIORITY	3740
6.776.1.24	HEADER_LOGIN	3740
6.776.1.25	HEADER_MAXPENDINGMSGLIMIT	3740
6.776.1.26	HEADER_MESSAGE	3740
6.776.1.27	HEADER_MESSAGEID	3740
6.776.1.28	HEADER_NOLOCAL	3740
6.776.1.29	HEADER_OLDSUBSCRIPTIONNAME	3740
6.776.1.30	HEADER_PASSWORD	3740
6.776.1.31	HEADER_PERSISTENT	3740
6.776.1.32	HEADER_PREFETCHSIZE	3740
6.776.1.33	HEADER_RECEIPT_REQUIRED	3740
6.776.1.34	HEADER_RECEIPTID	3740
6.776.1.35	HEADER_REDELIVERED	3740
6.776.1.36	HEADER_REDELIVERYCOUNT	3740
6.776.1.37	HEADER_REPLYTO	3740
6.776.1.38	HEADER_REQUESTID	3740
6.776.1.39	HEADER_RESPONSEID	3740
6.776.1.40	HEADER_RETROACTIVE	3740
6.776.1.41	HEADER_SELECTOR	3740
6.776.1.42	HEADER_SESSIONID	3740
6.776.1.43	HEADER_SUBSCRIPTION	3740
6.776.1.44	HEADER_SUBSCRIPTIONNAME	3740
6.776.1.45	HEADER_TIMESTAMP	3740
6.776.1.46	HEADER_TRANSACTIONID	3740
6.776.1.47	HEADER_TRANSFORMATION	3740
6.776.1.48	HEADER_TRANSFORMATION_ERROR	3740
6.776.1.49	HEADER_TYPE	3740
6.776.1.50	MESSAGE	3740
6.776.1.51	QUEUE_PREFIX	3740
6.776.1.52	RECEIPT	3740
6.776.1.53	SEND	3740
6.776.1.54	SUBSCRIBE	3740
6.776.1.55	TEMPQUEUE_PREFIX	3740
6.776.1.56	TEMPTOPIC_PREFIX	3740
6.776.1.57	TEXT	3740
6.776.1.58	TOPIC_PREFIX	3740
6.776.1.59	UNSUBSCRIBE	3740
6.777	activemq::wireformat::stomp::StompFrame Class Reference	3741
6.777.1	Detailed Description	3742
6.777.2	Constructor & Destructor Documentation	3742
6.777.2.1	StompFrame	3742
6.777.2.2	~StompFrame	3742
6.777.3	Member Function Documentation	3742
6.777.3.1	clone	3742
6.777.3.2	copy	3743
6.777.3.3	fromStream	3743
6.777.3.4	getBody	3743



6.777.3.5	getBody	3743
6.777.3.6	getBodyLength	3743
6.777.3.7	getCommand	3744
6.777.3.8	getProperties	3744
6.777.3.9	getProperties	3744
6.777.3.10	getProperty	3744
6.777.3.11	hasProperty	3744
6.777.3.12	removeProperty	3745
6.777.3.13	setBody	3745
6.777.3.14	setCommand	3745
6.777.3.15	setProperty	3745
6.777.3.16	toStream	3745
6.778	activemq::wireformat::stomp::StompHelper Class Reference	3746
6.778.1	Detailed Description	3747
6.778.2	Constructor & Destructor Documentation	3747
6.778.2.1	StompHelper	3747
6.778.2.2	~StompHelper	3747
6.778.3	Member Function Documentation	3747
6.778.3.1	convertConsumerId	3747
6.778.3.2	convertConsumerId	3748
6.778.3.3	convertDestination	3748
6.778.3.4	convertDestination	3748
6.778.3.5	convertMessageId	3748
6.778.3.6	convertMessageId	3749
6.778.3.7	convertProducerId	3749
6.778.3.8	convertProducerId	3749
6.778.3.9	convertProperties	3750
6.778.3.10	convertProperties	3750
6.778.3.11	convertTransactionId	3750
6.778.3.12	convertTransactionId	3750
6.779	activemq::wireformat::stomp::StompWireFormat Class Reference	3751
6.779.1	Constructor & Destructor Documentation	3752
6.779.1.1	StompWireFormat	3752
6.779.1.2	~StompWireFormat	3752
6.779.2	Member Function Documentation	3752
6.779.2.1	createNegotiator	3752
6.779.2.2	getVersion	3752
6.779.2.3	hasNegotiator	3753
6.779.2.4	inReceive	3753
6.779.2.5	marshal	3753
6.779.2.6	setVersion	3753
6.779.2.7	unmarshal	3754
6.780	activemq::wireformat::stomp::StompWireFormatFactory Class Reference	3754
6.780.1	Detailed Description	3755
6.780.2	Constructor & Destructor Documentation	3755
6.780.2.1	StompWireFormatFactory	3755
6.780.2.2	~StompWireFormatFactory	3755
6.780.3	Member Function Documentation	3755
6.780.3.1	createWireFormat	3755

6.781cms::Stoppable Class Reference . . . . .	3755
6.781.1 Detailed Description . . . . .	3756
6.781.2 Constructor & Destructor Documentation . . . . .	3756
6.781.2.1 ~Stoppable . . . . .	3756
6.781.3 Member Function Documentation . . . . .	3756
6.781.3.1 stop . . . . .	3756
6.782decaf::util::logging::StreamHandler Class Reference . . . . .	3756
6.782.1 Detailed Description . . . . .	3757
6.782.2 Constructor & Destructor Documentation . . . . .	3758
6.782.2.1 StreamHandler . . . . .	3758
6.782.2.2 StreamHandler . . . . .	3758
6.782.2.3 ~StreamHandler . . . . .	3758
6.782.3 Member Function Documentation . . . . .	3758
6.782.3.1 close . . . . .	3758
6.782.3.2 close . . . . .	3758
6.782.3.3 flush . . . . .	3759
6.782.3.4 isLoggable . . . . .	3759
6.782.3.5 publish . . . . .	3759
6.782.3.6 setOutputStream . . . . .	3759
6.783cms::StreamMessage Class Reference . . . . .	3760
6.783.1 Detailed Description . . . . .	3762
6.783.2 Constructor & Destructor Documentation . . . . .	3763
6.783.2.1 ~StreamMessage . . . . .	3763
6.783.3 Member Function Documentation . . . . .	3763
6.783.3.1 readBoolean . . . . .	3763
6.783.3.2 readByte . . . . .	3764
6.783.3.3 readBytes . . . . .	3764
6.783.3.4 readBytes . . . . .	3765
6.783.3.5 readChar . . . . .	3766
6.783.3.6 readDouble . . . . .	3766
6.783.3.7 readFloat . . . . .	3767
6.783.3.8 readInt . . . . .	3767
6.783.3.9 readLong . . . . .	3768
6.783.3.10readShort . . . . .	3769
6.783.3.11readString . . . . .	3769
6.783.3.12readUnsignedShort . . . . .	3770
6.783.3.13writeBoolean . . . . .	3770
6.783.3.14writeByte . . . . .	3771
6.783.3.15writeBytes . . . . .	3771
6.783.3.16writeBytes . . . . .	3772
6.783.3.17writeChar . . . . .	3772
6.783.3.18writeDouble . . . . .	3772
6.783.3.19writeFloat . . . . .	3773
6.783.3.20writeInt . . . . .	3773
6.783.3.21writeLong . . . . .	3774
6.783.3.22writeShort . . . . .	3774
6.783.3.23writeString . . . . .	3774
6.783.3.24writeUnsignedShort . . . . .	3775
6.784decaf::lang::String Class Reference . . . . .	3775
6.784.1 Detailed Description . . . . .	3777

6.784.2 Constructor & Destructor Documentation . . . . .	3777
6.784.2.1 String . . . . .	3777
6.784.2.2 String . . . . .	3777
6.784.2.3 ~String . . . . .	3777
6.784.3 Member Function Documentation . . . . .	3777
6.784.3.1 charAt . . . . .	3777
6.784.3.2 isEmpty . . . . .	3778
6.784.3.3 length . . . . .	3778
6.784.3.4 subSequence . . . . .	3778
6.784.3.5 toString . . . . .	3778
6.785decaf::util::StringTokenizer Class Reference . . . . .	3779
6.785.1 Constructor & Destructor Documentation . . . . .	3779
6.785.1.1 StringTokenizer . . . . .	3779
6.785.1.2 ~StringTokenizer . . . . .	3780
6.785.2 Member Function Documentation . . . . .	3780
6.785.2.1 countTokens . . . . .	3780
6.785.2.2 hasMoreTokens . . . . .	3780
6.785.2.3 nextToken . . . . .	3780
6.785.2.4 nextToken . . . . .	3781
6.785.2.5 reset . . . . .	3781
6.785.2.6 toArray . . . . .	3781
6.786activemq::commands::SubscriptionInfo Class Reference . . . . .	3782
6.786.1 Constructor & Destructor Documentation . . . . .	3783
6.786.1.1 SubscriptionInfo . . . . .	3783
6.786.1.2 ~SubscriptionInfo . . . . .	3783
6.786.2 Member Function Documentation . . . . .	3783
6.786.2.1 cloneDataStructure . . . . .	3783
6.786.2.2 copyDataStructure . . . . .	3784
6.786.2.3 equals . . . . .	3784
6.786.2.4 getClientId . . . . .	3784
6.786.2.5 getClientId . . . . .	3784
6.786.2.6 getDataStructureType . . . . .	3784
6.786.2.7 getDestination . . . . .	3785
6.786.2.8 getDestination . . . . .	3785
6.786.2.9 getSelector . . . . .	3785
6.786.2.10getSelector . . . . .	3785
6.786.2.11getSubscriptionName . . . . .	3785
6.786.2.12getSubscriptionName . . . . .	3785
6.786.2.13getSubscribedDestination . . . . .	3785
6.786.2.14getSubscribedDestination . . . . .	3785
6.786.2.15setClientId . . . . .	3785
6.786.2.16setDestination . . . . .	3785
6.786.2.17setSelector . . . . .	3785
6.786.2.18setSubscriptionName . . . . .	3785
6.786.2.19setSubscribedDestination . . . . .	3785
6.786.2.20toString . . . . .	3785
6.786.3 Field Documentation . . . . .	3786
6.786.3.1 clientId . . . . .	3786
6.786.3.2 destination . . . . .	3786
6.786.3.3 ID_SUBSCRIPTIONINFO . . . . .	3786

6.786.3.4 selector . . . . .	3786
6.786.3.5 subscriptionName . . . . .	3786
6.786.3.6 subscribedDestination . . . . .	3786
6.787activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	
Class Reference . . . . .	3786
6.787.1 Detailed Description . . . . .	3787
6.787.2 Constructor & Destructor Documentation . . . . .	3788
6.787.2.1 SubscriptionInfoMarshaller . . . . .	3788
6.787.2.2 ~SubscriptionInfoMarshaller . . . . .	3788
6.787.3 Member Function Documentation . . . . .	3788
6.787.3.1 createObject . . . . .	3788
6.787.3.2 getDataStructureType . . . . .	3788
6.787.3.3 looseMarshal . . . . .	3788
6.787.3.4 looseUnmarshal . . . . .	3789
6.787.3.5 tightMarshal1 . . . . .	3789
6.787.3.6 tightMarshal2 . . . . .	3790
6.787.3.7 tightUnmarshal . . . . .	3790
6.788activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	
Class Reference . . . . .	3791
6.788.1 Detailed Description . . . . .	3792
6.788.2 Constructor & Destructor Documentation . . . . .	3792
6.788.2.1 SubscriptionInfoMarshaller . . . . .	3792
6.788.2.2 ~SubscriptionInfoMarshaller . . . . .	3792
6.788.3 Member Function Documentation . . . . .	3792
6.788.3.1 createObject . . . . .	3792
6.788.3.2 getDataStructureType . . . . .	3792
6.788.3.3 looseMarshal . . . . .	3792
6.788.3.4 looseUnmarshal . . . . .	3793
6.788.3.5 tightMarshal1 . . . . .	3793
6.788.3.6 tightMarshal2 . . . . .	3794
6.788.3.7 tightUnmarshal . . . . .	3794
6.789activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller	
Class Reference . . . . .	3795
6.789.1 Detailed Description . . . . .	3796
6.789.2 Constructor & Destructor Documentation . . . . .	3796
6.789.2.1 SubscriptionInfoMarshaller . . . . .	3796
6.789.2.2 ~SubscriptionInfoMarshaller . . . . .	3796
6.789.3 Member Function Documentation . . . . .	3796
6.789.3.1 createObject . . . . .	3796
6.789.3.2 getDataStructureType . . . . .	3796
6.789.3.3 looseMarshal . . . . .	3796
6.789.3.4 looseUnmarshal . . . . .	3797
6.789.3.5 tightMarshal1 . . . . .	3797
6.789.3.6 tightMarshal2 . . . . .	3798
6.789.3.7 tightUnmarshal . . . . .	3798
6.790activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	
Class Reference . . . . .	3799
6.790.1 Detailed Description . . . . .	3800
6.790.2 Constructor & Destructor Documentation . . . . .	3800
6.790.2.1 SubscriptionInfoMarshaller . . . . .	3800

6.790.2.2	~SubscriptionInfoMarshaller	3800
6.790.3	Member Function Documentation	3800
6.790.3.1	createObject	3800
6.790.3.2	getDataStructureType	3800
6.790.3.3	looseMarshal	3800
6.790.3.4	looseUnmarshal	3801
6.790.3.5	tightMarshal1	3801
6.790.3.6	tightMarshal2	3802
6.790.3.7	tightUnmarshal	3802
6.791	activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller	
	Class Reference	3803
6.791.1	Detailed Description	3804
6.791.2	Constructor & Destructor Documentation	3804
6.791.2.1	SubscriptionInfoMarshaller	3804
6.791.2.2	~SubscriptionInfoMarshaller	3804
6.791.3	Member Function Documentation	3804
6.791.3.1	createObject	3804
6.791.3.2	getDataStructureType	3804
6.791.3.3	looseMarshal	3804
6.791.3.4	looseUnmarshal	3805
6.791.3.5	tightMarshal1	3805
6.791.3.6	tightMarshal2	3806
6.791.3.7	tightUnmarshal	3806
6.792	activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller	
	Class Reference	3807
6.792.1	Detailed Description	3808
6.792.2	Constructor & Destructor Documentation	3808
6.792.2.1	SubscriptionInfoMarshaller	3808
6.792.2.2	~SubscriptionInfoMarshaller	3808
6.792.3	Member Function Documentation	3808
6.792.3.1	createObject	3808
6.792.3.2	getDataStructureType	3808
6.792.3.3	looseMarshal	3808
6.792.3.4	looseUnmarshal	3809
6.792.3.5	tightMarshal1	3809
6.792.3.6	tightMarshal2	3810
6.792.3.7	tightUnmarshal	3810
6.793	decaf::util::concurrent::Synchronizable Class Reference	3811
6.793.1	Detailed Description	3812
6.793.2	Constructor & Destructor Documentation	3812
6.793.2.1	~Synchronizable	3812
6.793.3	Member Function Documentation	3812
6.793.3.1	lock	3812
6.793.3.2	notify	3813
6.793.3.3	notifyAll	3814
6.793.3.4	tryLock	3815
6.793.3.5	unlock	3816
6.793.3.6	wait	3818
6.793.3.7	wait	3819
6.793.3.8	wait	3820

6.794	decaf::internal::util::concurrent::SynchronizableImpl Class Reference	3822
6.794.1	Detailed Description	3823
6.794.2	Constructor & Destructor Documentation	3823
6.794.2.1	SynchronizableImpl	3823
6.794.2.2	~SynchronizableImpl	3823
6.794.3	Member Function Documentation	3823
6.794.3.1	lock	3823
6.794.3.2	notify	3823
6.794.3.3	notifyAll	3824
6.794.3.4	tryLock	3824
6.794.3.5	unlock	3824
6.794.3.6	wait	3825
6.794.3.7	wait	3825
6.794.3.8	wait	3825
6.795	activemq::core::Synchronization Class Reference	3826
6.795.1	Detailed Description	3826
6.795.2	Constructor & Destructor Documentation	3827
6.795.2.1	~Synchronization	3827
6.795.3	Member Function Documentation	3827
6.795.3.1	afterCommit	3827
6.795.3.2	afterRollback	3827
6.795.3.3	beforeEnd	3827
6.796	decaf::util::concurrent::SynchronousQueue< E > Class Template Reference	3827
6.796.1	Detailed Description	3829
6.796.2	Constructor & Destructor Documentation	3830
6.796.2.1	SynchronousQueue	3830
6.796.2.2	~SynchronousQueue	3830
6.796.3	Member Function Documentation	3830
6.796.3.1	clear	3830
6.796.3.2	contains	3830
6.796.3.3	containsAll	3830
6.796.3.4	drainTo	3831
6.796.3.5	drainTo	3832
6.796.3.6	equals	3832
6.796.3.7	isEmpty	3833
6.796.3.8	iterator	3833
6.796.3.9	iterator	3833
6.796.3.10	offer	3833
6.796.3.11	offer	3834
6.796.3.12	peek	3834
6.796.3.13	poll	3834
6.796.3.14	poll	3835
6.796.3.15	put	3835
6.796.3.16	remainingCapacity	3836
6.796.3.17	remove	3837
6.796.3.18	removeAll	3837
6.796.3.19	retainAll	3837
6.796.3.20	size	3837
6.796.3.21	take	3837

6.796.3.22	toArray . . . . .	3838
6.797	decaf::lang::System Class Reference . . . . .	3838
6.797.1	Detailed Description . . . . .	3840
6.797.2	Constructor & Destructor Documentation . . . . .	3840
6.797.2.1	System . . . . .	3840
6.797.2.2	~System . . . . .	3840
6.797.3	Member Function Documentation . . . . .	3840
6.797.3.1	arraycopy . . . . .	3840
6.797.3.2	arraycopy . . . . .	3841
6.797.3.3	arraycopy . . . . .	3841
6.797.3.4	arraycopy . . . . .	3842
6.797.3.5	availableProcessors . . . . .	3842
6.797.3.6	clearProperty . . . . .	3842
6.797.3.7	currentTimeMillis . . . . .	3843
6.797.3.8	getenv . . . . .	3843
6.797.3.9	getenv . . . . .	3843
6.797.3.10	getProperties . . . . .	3844
6.797.3.11	getProperty . . . . .	3844
6.797.3.12	getProperty . . . . .	3844
6.797.3.13	nanoTime . . . . .	3845
6.797.3.14	setenv . . . . .	3845
6.797.3.15	setProperty . . . . .	3846
6.797.3.16	unsetenv . . . . .	3846
6.797.4	Friends And Related Function Documentation . . . . .	3846
6.797.4.1	decaf::lang::Runtime . . . . .	3846
6.798	activemq::threads::Task Class Reference . . . . .	3846
6.798.1	Detailed Description . . . . .	3847
6.798.2	Constructor & Destructor Documentation . . . . .	3847
6.798.2.1	~Task . . . . .	3847
6.798.3	Member Function Documentation . . . . .	3847
6.798.3.1	iterate . . . . .	3847
6.799	decaf::util::concurrent::TaskListener Class Reference . . . . .	3847
6.799.1	Constructor & Destructor Documentation . . . . .	3848
6.799.1.1	~TaskListener . . . . .	3848
6.799.2	Member Function Documentation . . . . .	3848
6.799.2.1	onTaskComplete . . . . .	3848
6.799.2.2	onTaskException . . . . .	3848
6.800	activemq::threads::TaskRunner Class Reference . . . . .	3849
6.800.1	Constructor & Destructor Documentation . . . . .	3849
6.800.1.1	~TaskRunner . . . . .	3849
6.800.2	Member Function Documentation . . . . .	3849
6.800.2.1	shutdown . . . . .	3849
6.800.2.2	shutdown . . . . .	3849
6.800.2.3	wakeup . . . . .	3850
6.801	decaf::internal::net::tcp::TcpSocket Class Reference . . . . .	3850
6.801.1	Detailed Description . . . . .	3853
6.801.2	Constructor & Destructor Documentation . . . . .	3854
6.801.2.1	TcpSocket . . . . .	3854
6.801.2.2	~TcpSocket . . . . .	3854
6.801.3	Member Function Documentation . . . . .	3854

6.801.3.1	accept . . . . .	3854
6.801.3.2	available . . . . .	3854
6.801.3.3	bind . . . . .	3854
6.801.3.4	checkResult . . . . .	3855
6.801.3.5	close . . . . .	3855
6.801.3.6	connect . . . . .	3855
6.801.3.7	create . . . . .	3855
6.801.3.8	getInputStream . . . . .	3856
6.801.3.9	getLocalAddress . . . . .	3856
6.801.3.10	getOption . . . . .	3856
6.801.3.11	getOutputStream . . . . .	3857
6.801.3.12	getSocketHandle . . . . .	3857
6.801.3.13	isClosed . . . . .	3857
6.801.3.14	isConnected . . . . .	3857
6.801.3.15	listen . . . . .	3857
6.801.3.16	read . . . . .	3858
6.801.3.17	setOption . . . . .	3858
6.801.3.18	shutdownInput . . . . .	3859
6.801.3.19	shutdownOutput . . . . .	3859
6.801.3.20	write . . . . .	3859
6.802	decaf::internal::net::tcp::TcpSocketInputStream Class Reference . . .	3860
6.802.1	Detailed Description . . . . .	3861
6.802.2	Constructor & Destructor Documentation . . . . .	3861
6.802.2.1	TcpSocketInputStream . . . . .	3861
6.802.2.2	~TcpSocketInputStream . . . . .	3861
6.802.3	Member Function Documentation . . . . .	3861
6.802.3.1	available . . . . .	3861
6.802.3.2	close . . . . .	3862
6.802.3.3	doReadArrayBounded . . . . .	3862
6.802.3.4	doReadByte . . . . .	3862
6.802.3.5	skip . . . . .	3862
6.803	decaf::internal::net::tcp::TcpSocketOutputStream Class Reference . .	3863
6.803.1	Detailed Description . . . . .	3864
6.803.2	Constructor & Destructor Documentation . . . . .	3864
6.803.2.1	TcpSocketOutputStream . . . . .	3864
6.803.2.2	~TcpSocketOutputStream . . . . .	3864
6.803.3	Member Function Documentation . . . . .	3864
6.803.3.1	close . . . . .	3864
6.803.3.2	doWriteArrayBounded . . . . .	3865
6.803.3.3	doWriteByte . . . . .	3865
6.804	activemq::transport::tcp::TcpTransport Class Reference . . . . .	3865
6.804.1	Detailed Description . . . . .	3866
6.804.2	Constructor & Destructor Documentation . . . . .	3866
6.804.2.1	TcpTransport . . . . .	3866
6.804.2.2	~TcpTransport . . . . .	3867
6.804.3	Member Function Documentation . . . . .	3867
6.804.3.1	close . . . . .	3867
6.804.3.2	configureSocket . . . . .	3867
6.804.3.3	connect . . . . .	3867
6.804.3.4	createSocket . . . . .	3868



6.804.3.5	isClosed	3868
6.804.3.6	isConnected	3868
6.804.3.7	isFaultTolerant	3868
6.805	activemq::transport::tcp::TcpTransportFactory Class Reference	3869
6.805.1	Detailed Description	3869
6.805.2	Constructor & Destructor Documentation	3870
6.805.2.1	~TcpTransportFactory	3870
6.805.3	Member Function Documentation	3870
6.805.3.1	create	3870
6.805.3.2	createComposite	3870
6.805.3.3	doCreateComposite	3871
6.806	cms::TemporaryQueue Class Reference	3871
6.806.1	Detailed Description	3872
6.806.2	Constructor & Destructor Documentation	3872
6.806.2.1	~TemporaryQueue	3872
6.806.3	Member Function Documentation	3872
6.806.3.1	destroy	3872
6.806.3.2	getQueueName	3872
6.807	cms::TemporaryTopic Class Reference	3873
6.807.1	Detailed Description	3873
6.807.2	Constructor & Destructor Documentation	3873
6.807.2.1	~TemporaryTopic	3873
6.807.3	Member Function Documentation	3873
6.807.3.1	destroy	3873
6.807.3.2	getTopicName	3874
6.808	cms::TextMessage Class Reference	3874
6.808.1	Detailed Description	3875
6.808.2	Constructor & Destructor Documentation	3875
6.808.2.1	~TextMessage	3875
6.808.3	Member Function Documentation	3875
6.808.3.1	getText	3875
6.808.3.2	setText	3875
6.808.3.3	setText	3876
6.809	decaf::lang::Thread Class Reference	3876
6.809.1	Detailed Description	3879
6.809.2	Member Enumeration Documentation	3880
6.809.2.1	State	3880
6.809.3	Constructor & Destructor Documentation	3880
6.809.3.1	Thread	3880
6.809.3.2	Thread	3880
6.809.3.3	Thread	3881
6.809.3.4	Thread	3881
6.809.3.5	~Thread	3881
6.809.4	Member Function Documentation	3881
6.809.4.1	currentThread	3881
6.809.4.2	getId	3882
6.809.4.3	getName	3882
6.809.4.4	getPriority	3882
6.809.4.5	getState	3882
6.809.4.6	getUncaughtExceptionHandler	3882

6.809.4.7	isAlive . . . . .	3883
6.809.4.8	join . . . . .	3883
6.809.4.9	join . . . . .	3883
6.809.4.10	join . . . . .	3883
6.809.4.11	run . . . . .	3884
6.809.4.12	setName . . . . .	3884
6.809.4.13	setPriority . . . . .	3884
6.809.4.14	setUncaughtExceptionHandler . . . . .	3884
6.809.4.15	sleep . . . . .	3885
6.809.4.16	sleep . . . . .	3885
6.809.4.17	start . . . . .	3885
6.809.4.18	toString . . . . .	3886
6.809.4.19	yield . . . . .	3886
6.809.5	Friends And Related Function Documentation . . . . .	3886
6.809.5.1	decaf::lang::Runtime . . . . .	3886
6.809.5.2	decaf::util::concurrent::locks::LockSupport . . . . .	3886
6.809.6	Field Documentation . . . . .	3886
6.809.6.1	MAX_PRIORITY . . . . .	3886
6.809.6.2	MIN_PRIORITY . . . . .	3886
6.809.6.3	NORM_PRIORITY . . . . .	3886
6.810	decaf::util::concurrent::ThreadFactory Class Reference . . . . .	3887
6.810.1	Detailed Description . . . . .	3887
6.810.2	Constructor & Destructor Documentation . . . . .	3887
6.810.2.1	~ThreadFactory . . . . .	3887
6.810.3	Member Function Documentation . . . . .	3887
6.810.3.1	newThread . . . . .	3887
6.811	decaf::lang::ThreadGroup Class Reference . . . . .	3888
6.811.1	Detailed Description . . . . .	3888
6.811.2	Constructor & Destructor Documentation . . . . .	3888
6.811.2.1	ThreadGroup . . . . .	3888
6.811.2.2	~ThreadGroup . . . . .	3888
6.812	decaf::util::concurrent::ThreadPool Class Reference . . . . .	3888
6.812.1	Detailed Description . . . . .	3890
6.812.2	Member Typedef Documentation . . . . .	3891
6.812.2.1	Task . . . . .	3891
6.812.3	Constructor & Destructor Documentation . . . . .	3891
6.812.3.1	ThreadPool . . . . .	3891
6.812.3.2	~ThreadPool . . . . .	3891
6.812.4	Member Function Documentation . . . . .	3891
6.812.4.1	deQueueTask . . . . .	3891
6.812.4.2	getBacklog . . . . .	3891
6.812.4.3	getBlockSize . . . . .	3891
6.812.4.4	getFreeThreadCount . . . . .	3892
6.812.4.5	getInstance . . . . .	3892
6.812.4.6	getMaxThreads . . . . .	3892
6.812.4.7	getPoolSize . . . . .	3892
6.812.4.8	onTaskCompleted . . . . .	3892
6.812.4.9	onTaskException . . . . .	3893
6.812.4.10	onTaskStarted . . . . .	3893
6.812.4.11	lqueueTask . . . . .	3893

6.812.4.12	reserve . . . . .	3894
6.812.4.13	setBlockSize . . . . .	3894
6.812.4.14	setMaxThreads . . . . .	3894
6.812.5	Field Documentation . . . . .	3894
6.812.5.1	DEFAULT_MAX_BLOCK_SIZE . . . . .	3894
6.812.5.2	DEFAULT_MAX_POOL_SIZE . . . . .	3894
6.813	decaf::lang::Throwable Class Reference . . . . .	3895
6.813.1	Detailed Description . . . . .	3896
6.813.2	Constructor & Destructor Documentation . . . . .	3896
6.813.2.1	Throwable . . . . .	3896
6.813.2.2	~Throwable . . . . .	3896
6.813.3	Member Function Documentation . . . . .	3896
6.813.3.1	clone . . . . .	3896
6.813.3.2	getCause . . . . .	3897
6.813.3.3	getMessage . . . . .	3897
6.813.3.4	getStackTrace . . . . .	3897
6.813.3.5	getStackTraceString . . . . .	3898
6.813.3.6	initCause . . . . .	3898
6.813.3.7	printStackTrace . . . . .	3898
6.813.3.8	printStackTrace . . . . .	3898
6.813.3.9	setMark . . . . .	3899
6.814	decaf::util::concurrent::TimeoutException Class Reference . . . . .	3899
6.814.1	Constructor & Destructor Documentation . . . . .	3900
6.814.1.1	TimeoutException . . . . .	3900
6.814.1.2	TimeoutException . . . . .	3900
6.814.1.3	TimeoutException . . . . .	3900
6.814.1.4	TimeoutException . . . . .	3900
6.814.1.5	TimeoutException . . . . .	3900
6.814.1.6	TimeoutException . . . . .	3901
6.814.1.7	~TimeoutException . . . . .	3901
6.814.2	Member Function Documentation . . . . .	3901
6.814.2.1	clone . . . . .	3901
6.815	decaf::util::Timer Class Reference . . . . .	3901
6.815.1	Detailed Description . . . . .	3903
6.815.2	Constructor & Destructor Documentation . . . . .	3904
6.815.2.1	Timer . . . . .	3904
6.815.2.2	~Timer . . . . .	3904
6.815.3	Member Function Documentation . . . . .	3904
6.815.3.1	cancel . . . . .	3904
6.815.3.2	purge . . . . .	3904
6.815.3.3	schedule . . . . .	3905
6.815.3.4	schedule . . . . .	3905
6.815.3.5	schedule . . . . .	3906
6.815.3.6	schedule . . . . .	3907
6.815.3.7	schedule . . . . .	3907
6.815.3.8	schedule . . . . .	3908
6.815.3.9	schedule . . . . .	3909
6.815.3.10	schedule . . . . .	3910
6.815.3.11	scheduleAtFixedRate . . . . .	3910
6.815.3.12	scheduleAtFixedRate . . . . .	3911

6.815.3.13	scheduleAtFixedRate	3912
6.815.3.14	scheduleAtFixedRate	3913
6.816	decaf::util::TimerTask Class Reference	3914
6.816.1	Detailed Description	3915
6.816.2	Constructor & Destructor Documentation	3915
6.816.2.1	TimerTask	3915
6.816.2.2	~TimerTask	3915
6.816.3	Member Function Documentation	3915
6.816.3.1	cancel	3915
6.816.3.2	getWhen	3916
6.816.3.3	isScheduled	3916
6.816.3.4	scheduledExecutionTime	3916
6.816.3.5	setScheduledTime	3916
6.816.4	Friends And Related Function Documentation	3916
6.816.4.1	decaf::internal::util::TimerTaskHeap	3916
6.816.4.2	Timer	3916
6.816.4.3	TimerImpl	3916
6.817	decaf::internal::util::TimerTaskHeap Class Reference	3916
6.817.1	Detailed Description	3917
6.817.2	Constructor & Destructor Documentation	3918
6.817.2.1	TimerTaskHeap	3918
6.817.2.2	~TimerTaskHeap	3918
6.817.3	Member Function Documentation	3918
6.817.3.1	adjustMinimum	3918
6.817.3.2	deleteIfCancelled	3918
6.817.3.3	find	3918
6.817.3.4	insert	3918
6.817.3.5	isEmpty	3919
6.817.3.6	peek	3919
6.817.3.7	remove	3919
6.817.3.8	reset	3919
6.817.3.9	size	3919
6.818	decaf::util::concurrent::TimeUnit Class Reference	3919
6.818.1	Detailed Description	3922
6.818.2	Constructor & Destructor Documentation	3922
6.818.2.1	TimeUnit	3922
6.818.2.2	~TimeUnit	3922
6.818.3	Member Function Documentation	3922
6.818.3.1	compareTo	3922
6.818.3.2	convert	3923
6.818.3.3	equals	3924
6.818.3.4	operator<	3924
6.818.3.5	operator==	3924
6.818.3.6	sleep	3924
6.818.3.7	timedJoin	3925
6.818.3.8	timedWait	3925
6.818.3.9	toDays	3926
6.818.3.10	toHours	3926
6.818.3.11	toMicros	3927
6.818.3.12	toMillis	3927

6.818.3.13	toMinutes	3927
6.818.3.14	toNanos	3928
6.818.3.15	toSeconds	3928
6.818.3.16	toString	3928
6.818.3.17	valueOf	3929
6.818.4	Field Documentation	3929
6.818.4.1	DAYS	3929
6.818.4.2	HOURS	3929
6.818.4.3	MICROSECONDS	3929
6.818.4.4	MILLISECONDS	3929
6.818.4.5	MINUTES	3929
6.818.4.6	NANOSECONDS	3929
6.818.4.7	SECONDS	3930
6.818.4.8	values	3930
6.819	cms::Topic Class Reference	3930
6.819.1	Detailed Description	3930
6.819.2	Constructor & Destructor Documentation	3931
6.819.2.1	~Topic	3931
6.819.3	Member Function Documentation	3931
6.819.3.1	getTopicName	3931
6.820	activemq::state::Tracked Class Reference	3931
6.820.1	Constructor & Destructor Documentation	3932
6.820.1.1	Tracked	3932
6.820.1.2	Tracked	3932
6.820.1.3	~Tracked	3932
6.820.2	Member Function Documentation	3932
6.820.2.1	isWaitingForResponse	3932
6.820.2.2	onResponse	3932
6.821	activemq::commands::TransactionId Class Reference	3932
6.821.1	Member Typedef Documentation	3933
6.821.1.1	COMPARATOR	3933
6.821.2	Constructor & Destructor Documentation	3933
6.821.2.1	TransactionId	3933
6.821.2.2	TransactionId	3933
6.821.2.3	~TransactionId	3933
6.821.3	Member Function Documentation	3933
6.821.3.1	cloneDataStructure	3933
6.821.3.2	compareTo	3934
6.821.3.3	copyDataStructure	3934
6.821.3.4	equals	3934
6.821.3.5	equals	3934
6.821.3.6	getDataStructureType	3934
6.821.3.7	operator<	3935
6.821.3.8	operator=	3935
6.821.3.9	operator==	3935
6.821.3.10	toString	3935
6.821.4	Field Documentation	3935
6.821.4.1	ID_TRANSACTIONID	3935
6.822	activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller	
	Class Reference	3935

6.822.1 Detailed Description . . . . .	3936
6.822.2 Constructor & Destructor Documentation . . . . .	3937
6.822.2.1 TransactionIdMarshaller . . . . .	3937
6.822.2.2 ~TransactionIdMarshaller . . . . .	3937
6.822.3 Member Function Documentation . . . . .	3937
6.822.3.1 looseMarshal . . . . .	3937
6.822.3.2 looseUnmarshal . . . . .	3937
6.822.3.3 tightMarshal1 . . . . .	3938
6.822.3.4 tightMarshal2 . . . . .	3938
6.822.3.5 tightUnmarshal . . . . .	3939
6.823activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller	
Class Reference . . . . .	3939
6.823.1 Detailed Description . . . . .	3940
6.823.2 Constructor & Destructor Documentation . . . . .	3941
6.823.2.1 TransactionIdMarshaller . . . . .	3941
6.823.2.2 ~TransactionIdMarshaller . . . . .	3941
6.823.3 Member Function Documentation . . . . .	3941
6.823.3.1 looseMarshal . . . . .	3941
6.823.3.2 looseUnmarshal . . . . .	3941
6.823.3.3 tightMarshal1 . . . . .	3942
6.823.3.4 tightMarshal2 . . . . .	3942
6.823.3.5 tightUnmarshal . . . . .	3943
6.824activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller	
Class Reference . . . . .	3943
6.824.1 Detailed Description . . . . .	3944
6.824.2 Constructor & Destructor Documentation . . . . .	3945
6.824.2.1 TransactionIdMarshaller . . . . .	3945
6.824.2.2 ~TransactionIdMarshaller . . . . .	3945
6.824.3 Member Function Documentation . . . . .	3945
6.824.3.1 looseMarshal . . . . .	3945
6.824.3.2 looseUnmarshal . . . . .	3945
6.824.3.3 tightMarshal1 . . . . .	3946
6.824.3.4 tightMarshal2 . . . . .	3946
6.824.3.5 tightUnmarshal . . . . .	3947
6.825activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller	
Class Reference . . . . .	3947
6.825.1 Detailed Description . . . . .	3948
6.825.2 Constructor & Destructor Documentation . . . . .	3949
6.825.2.1 TransactionIdMarshaller . . . . .	3949
6.825.2.2 ~TransactionIdMarshaller . . . . .	3949
6.825.3 Member Function Documentation . . . . .	3949
6.825.3.1 looseMarshal . . . . .	3949
6.825.3.2 looseUnmarshal . . . . .	3949
6.825.3.3 tightMarshal1 . . . . .	3950
6.825.3.4 tightMarshal2 . . . . .	3950
6.825.3.5 tightUnmarshal . . . . .	3951
6.826activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller	
Class Reference . . . . .	3951
6.826.1 Detailed Description . . . . .	3952
6.826.2 Constructor & Destructor Documentation . . . . .	3953

6.826.2.1 TransactionIdMarshaller . . . . .	3953
6.826.2.2 ~TransactionIdMarshaller . . . . .	3953
6.826.3 Member Function Documentation . . . . .	3953
6.826.3.1 looseMarshal . . . . .	3953
6.826.3.2 looseUnmarshal . . . . .	3953
6.826.3.3 tightMarshal1 . . . . .	3954
6.826.3.4 tightMarshal2 . . . . .	3954
6.826.3.5 tightUnmarshal . . . . .	3955
6.827activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller	
Class Reference . . . . .	3955
6.827.1 Detailed Description . . . . .	3956
6.827.2 Constructor & Destructor Documentation . . . . .	3957
6.827.2.1 TransactionIdMarshaller . . . . .	3957
6.827.2.2 ~TransactionIdMarshaller . . . . .	3957
6.827.3 Member Function Documentation . . . . .	3957
6.827.3.1 looseMarshal . . . . .	3957
6.827.3.2 looseUnmarshal . . . . .	3957
6.827.3.3 tightMarshal1 . . . . .	3958
6.827.3.4 tightMarshal2 . . . . .	3958
6.827.3.5 tightUnmarshal . . . . .	3959
6.828activemq::commands::TransactionInfo Class Reference . . . . .	3959
6.828.1 Constructor & Destructor Documentation . . . . .	3961
6.828.1.1 TransactionInfo . . . . .	3961
6.828.1.2 ~TransactionInfo . . . . .	3961
6.828.2 Member Function Documentation . . . . .	3961
6.828.2.1 cloneDataStructure . . . . .	3961
6.828.2.2 copyDataStructure . . . . .	3961
6.828.2.3 equals . . . . .	3961
6.828.2.4 getConnectionId . . . . .	3962
6.828.2.5 getConnectionId . . . . .	3962
6.828.2.6 getDataStructureType . . . . .	3962
6.828.2.7 getTransactionId . . . . .	3962
6.828.2.8 getTransactionId . . . . .	3962
6.828.2.9 getType . . . . .	3962
6.828.2.10isTransactionInfo . . . . .	3962
6.828.2.11setConnectionId . . . . .	3963
6.828.2.12setTransactionId . . . . .	3963
6.828.2.13setType . . . . .	3963
6.828.2.14toString . . . . .	3963
6.828.2.15visit . . . . .	3963
6.828.3 Field Documentation . . . . .	3964
6.828.3.1 connectionId . . . . .	3964
6.828.3.2 ID_TRANSACTIONINFO . . . . .	3964
6.828.3.3 transactionId . . . . .	3964
6.828.3.4 type . . . . .	3964
6.829activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller	
Class Reference . . . . .	3964
6.829.1 Detailed Description . . . . .	3965
6.829.2 Constructor & Destructor Documentation . . . . .	3965
6.829.2.1 TransactionInfoMarshaller . . . . .	3965

6.829.2.2 ~TransactionInfoMarshaller . . . . .	3965
6.829.3 Member Function Documentation . . . . .	3965
6.829.3.1 createObject . . . . .	3965
6.829.3.2 getDataStructureType . . . . .	3966
6.829.3.3 looseMarshal . . . . .	3966
6.829.3.4 looseUnmarshal . . . . .	3966
6.829.3.5 tightMarshal1 . . . . .	3967
6.829.3.6 tightMarshal2 . . . . .	3967
6.829.3.7 tightUnmarshal . . . . .	3968
6.830activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller	
Class Reference . . . . .	3968
6.830.1 Detailed Description . . . . .	3969
6.830.2 Constructor & Destructor Documentation . . . . .	3969
6.830.2.1 TransactionInfoMarshaller . . . . .	3969
6.830.2.2 ~TransactionInfoMarshaller . . . . .	3969
6.830.3 Member Function Documentation . . . . .	3969
6.830.3.1 createObject . . . . .	3969
6.830.3.2 getDataStructureType . . . . .	3970
6.830.3.3 looseMarshal . . . . .	3970
6.830.3.4 looseUnmarshal . . . . .	3970
6.830.3.5 tightMarshal1 . . . . .	3971
6.830.3.6 tightMarshal2 . . . . .	3971
6.830.3.7 tightUnmarshal . . . . .	3972
6.831activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	
Class Reference . . . . .	3972
6.831.1 Detailed Description . . . . .	3973
6.831.2 Constructor & Destructor Documentation . . . . .	3974
6.831.2.1 TransactionInfoMarshaller . . . . .	3974
6.831.2.2 ~TransactionInfoMarshaller . . . . .	3974
6.831.3 Member Function Documentation . . . . .	3974
6.831.3.1 createObject . . . . .	3974
6.831.3.2 getDataStructureType . . . . .	3974
6.831.3.3 looseMarshal . . . . .	3974
6.831.3.4 looseUnmarshal . . . . .	3975
6.831.3.5 tightMarshal1 . . . . .	3975
6.831.3.6 tightMarshal2 . . . . .	3976
6.831.3.7 tightUnmarshal . . . . .	3976
6.832activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller	
Class Reference . . . . .	3977
6.832.1 Detailed Description . . . . .	3978
6.832.2 Constructor & Destructor Documentation . . . . .	3978
6.832.2.1 TransactionInfoMarshaller . . . . .	3978
6.832.2.2 ~TransactionInfoMarshaller . . . . .	3978
6.832.3 Member Function Documentation . . . . .	3978
6.832.3.1 createObject . . . . .	3978
6.832.3.2 getDataStructureType . . . . .	3978
6.832.3.3 looseMarshal . . . . .	3978
6.832.3.4 looseUnmarshal . . . . .	3979
6.832.3.5 tightMarshal1 . . . . .	3979
6.832.3.6 tightMarshal2 . . . . .	3980



6.832.3.7	tightUnmarshal . . . . .	3980
6.833	activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	
	Class Reference . . . . .	3981
6.833.1	Detailed Description . . . . .	3982
6.833.2	Constructor & Destructor Documentation . . . . .	3982
6.833.2.1	TransactionInfoMarshaller . . . . .	3982
6.833.2.2	~TransactionInfoMarshaller . . . . .	3982
6.833.3	Member Function Documentation . . . . .	3982
6.833.3.1	createObject . . . . .	3982
6.833.3.2	getDataStructureType . . . . .	3982
6.833.3.3	looseMarshal . . . . .	3983
6.833.3.4	looseUnmarshal . . . . .	3983
6.833.3.5	tightMarshal1 . . . . .	3984
6.833.3.6	tightMarshal2 . . . . .	3984
6.833.3.7	tightUnmarshal . . . . .	3985
6.834	activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller	
	Class Reference . . . . .	3985
6.834.1	Detailed Description . . . . .	3986
6.834.2	Constructor & Destructor Documentation . . . . .	3986
6.834.2.1	TransactionInfoMarshaller . . . . .	3986
6.834.2.2	~TransactionInfoMarshaller . . . . .	3986
6.834.3	Member Function Documentation . . . . .	3986
6.834.3.1	createObject . . . . .	3986
6.834.3.2	getDataStructureType . . . . .	3987
6.834.3.3	looseMarshal . . . . .	3987
6.834.3.4	looseUnmarshal . . . . .	3987
6.834.3.5	tightMarshal1 . . . . .	3988
6.834.3.6	tightMarshal2 . . . . .	3988
6.834.3.7	tightUnmarshal . . . . .	3989
6.835	activemq::state::TransactionState Class Reference . . . . .	3989
6.835.1	Constructor & Destructor Documentation . . . . .	3991
6.835.1.1	TransactionState . . . . .	3991
6.835.1.2	~TransactionState . . . . .	3991
6.835.2	Member Function Documentation . . . . .	3991
6.835.2.1	addCommand . . . . .	3991
6.835.2.2	addProducerState . . . . .	3991
6.835.2.3	checkShutdown . . . . .	3991
6.835.2.4	getCommands . . . . .	3991
6.835.2.5	getId . . . . .	3991
6.835.2.6	getPreparedResult . . . . .	3991
6.835.2.7	getProducerStates . . . . .	3991
6.835.2.8	isPrepared . . . . .	3991
6.835.2.9	setPrepared . . . . .	3991
6.835.2.10	setPreparedResult . . . . .	3991
6.835.2.11	shutdown . . . . .	3991
6.835.2.12	toString . . . . .	3991
6.836	decaf::internal::util::concurrent::Transferer< E > Class Template Reference . . . . .	3991
6.836.1	Detailed Description . . . . .	3992

6.837	decaf::internal::util::concurrent::TransferQueue< E > Class Template	
	Reference	3992
6.837.1	Detailed Description	3993
6.837.2	Constructor & Destructor Documentation	3993
6.837.2.1	TransferQueue	3993
6.837.2.2	~TransferQueue	3993
6.837.3	Member Function Documentation	3993
6.837.3.1	transfer	3993
6.837.3.2	transfer	3994
6.838	decaf::internal::util::concurrent::TransferStack< E > Class Template	
	Reference	3994
6.838.1	Constructor & Destructor Documentation	3995
6.838.1.1	TransferStack	3995
6.838.1.2	~TransferStack	3995
6.838.2	Member Function Documentation	3995
6.838.2.1	transfer	3995
6.838.2.2	transfer	3995
6.839	activemq::transport::Transport Class Reference	3996
6.839.1	Detailed Description	3997
6.839.2	Constructor & Destructor Documentation	3998
6.839.2.1	~Transport	3998
6.839.3	Member Function Documentation	3998
6.839.3.1	getRemoteAddress	3998
6.839.3.2	getTransportListener	3998
6.839.3.3	isClosed	3998
6.839.3.4	isConnected	3998
6.839.3.5	isFaultTolerant	3999
6.839.3.6	narrow	3999
6.839.3.7	oneway	3999
6.839.3.8	reconnect	4000
6.839.3.9	request	4000
6.839.3.10	request	4001
6.839.3.11	setTransportListener	4001
6.839.3.12	setWireFormat	4002
6.839.3.13	start	4002
6.839.3.14	stop	4002
6.840	activemq::transport::TransportFactory Class Reference	4003
6.840.1	Detailed Description	4003
6.840.2	Constructor & Destructor Documentation	4003
6.840.2.1	~TransportFactory	4003
6.840.3	Member Function Documentation	4003
6.840.3.1	create	4003
6.840.3.2	createComposite	4004
6.841	activemq::transport::TransportFilter Class Reference	4004
6.841.1	Detailed Description	4006
6.841.2	Constructor & Destructor Documentation	4007
6.841.2.1	TransportFilter	4007
6.841.2.2	~TransportFilter	4007
6.841.3	Member Function Documentation	4007
6.841.3.1	close	4007

6.841.3.2	fire	4007
6.841.3.3	fire	4008
6.841.3.4	getRemoteAddress	4008
6.841.3.5	getTransportListener	4008
6.841.3.6	isClosed	4008
6.841.3.7	isConnected	4008
6.841.3.8	isFaultTolerant	4009
6.841.3.9	narrow	4009
6.841.3.10	onCommand	4009
6.841.3.11	oneway	4010
6.841.3.12	onException	4010
6.841.3.13	reconnect	4010
6.841.3.14	request	4011
6.841.3.15	request	4011
6.841.3.16	setTransportListener	4012
6.841.3.17	setWireFormat	4012
6.841.3.18	start	4012
6.841.3.19	stop	4012
6.841.3.20	transportInterrupted	4013
6.841.3.21	transportResumed	4013
6.841.4	Field Documentation	4013
6.841.4.1	listener	4013
6.841.4.2	next	4013
6.842	activemq::transport::TransportListener Class Reference	4013
6.842.1	Detailed Description	4014
6.842.2	Constructor & Destructor Documentation	4014
6.842.2.1	~TransportListener	4014
6.842.3	Member Function Documentation	4014
6.842.3.1	onCommand	4014
6.842.3.2	onException	4015
6.842.3.3	transportInterrupted	4015
6.842.3.4	transportResumed	4015
6.843	activemq::transport::TransportRegistry Class Reference	4015
6.843.1	Detailed Description	4016
6.843.2	Constructor & Destructor Documentation	4016
6.843.2.1	~TransportRegistry	4016
6.843.3	Member Function Documentation	4016
6.843.3.1	findFactory	4016
6.843.3.2	getInstance	4017
6.843.3.3	getTransportNames	4017
6.843.3.4	registerFactory	4017
6.843.3.5	unregisterFactory	4018
6.844	tree_desc_s Struct Reference	4018
6.844.1	Field Documentation	4018
6.844.1.1	dyn_tree	4018
6.844.1.2	max_code	4018
6.844.1.3	stat_desc	4018
6.845	decaf::lang::Thread::UncaughtExceptionHandler Class Reference	4018
6.845.1	Detailed Description	4019
6.845.2	Constructor & Destructor Documentation	4019

6.845.2.1 ~UncaughtExceptionHandler . . . . .	4019
6.845.3 Member Function Documentation . . . . .	4019
6.845.3.1 uncaughtException . . . . .	4019
6.846decaf::net::UnknownHostException Class Reference . . . . .	4019
6.846.1 Constructor & Destructor Documentation . . . . .	4020
6.846.1.1 UnknownHostException . . . . .	4020
6.846.1.2 UnknownHostException . . . . .	4020
6.846.1.3 UnknownHostException . . . . .	4021
6.846.1.4 UnknownHostException . . . . .	4021
6.846.1.5 UnknownHostException . . . . .	4021
6.846.1.6 UnknownHostException . . . . .	4021
6.846.1.7 ~UnknownHostException . . . . .	4022
6.846.2 Member Function Documentation . . . . .	4022
6.846.2.1 clone . . . . .	4022
6.847decaf::net::UnknownServiceException Class Reference . . . . .	4022
6.847.1 Constructor & Destructor Documentation . . . . .	4023
6.847.1.1 UnknownServiceException . . . . .	4023
6.847.1.2 UnknownServiceException . . . . .	4023
6.847.1.3 UnknownServiceException . . . . .	4023
6.847.1.4 UnknownServiceException . . . . .	4023
6.847.1.5 UnknownServiceException . . . . .	4024
6.847.1.6 UnknownServiceException . . . . .	4024
6.847.1.7 ~UnknownServiceException . . . . .	4024
6.847.2 Member Function Documentation . . . . .	4024
6.847.2.1 clone . . . . .	4024
6.848decaf::io::UnsupportedEncodingException Class Reference . . . . .	4025
6.848.1 Detailed Description . . . . .	4026
6.848.2 Constructor & Destructor Documentation . . . . .	4026
6.848.2.1 UnsupportedEncodingException . . . . .	4026
6.848.2.2 UnsupportedEncodingException . . . . .	4026
6.848.2.3 UnsupportedEncodingException . . . . .	4026
6.848.2.4 UnsupportedEncodingException . . . . .	4026
6.848.2.5 UnsupportedEncodingException . . . . .	4027
6.848.2.6 UnsupportedEncodingException . . . . .	4027
6.848.2.7 ~UnsupportedEncodingException . . . . .	4027
6.848.3 Member Function Documentation . . . . .	4027
6.848.3.1 clone . . . . .	4027
6.849decaf::lang::exceptions::UnsupportedOperationException Class Refer- ence . . . . .	4028
6.849.1 Constructor & Destructor Documentation . . . . .	4028
6.849.1.1 UnsupportedOperationException . . . . .	4028
6.849.1.2 UnsupportedOperationException . . . . .	4029
6.849.1.3 UnsupportedOperationException . . . . .	4029
6.849.1.4 UnsupportedOperationException . . . . .	4029
6.849.1.5 UnsupportedOperationException . . . . .	4029
6.849.1.6 UnsupportedOperationException . . . . .	4030
6.849.1.7 ~UnsupportedOperationException . . . . .	4030
6.849.2 Member Function Documentation . . . . .	4030
6.849.2.1 clone . . . . .	4030
6.850cms::UnsupportedOperationException Class Reference . . . . .	4030

6.850.1 Detailed Description . . . . .	4031
6.850.2 Constructor & Destructor Documentation . . . . .	4031
6.850.2.1 UnsupportedOperationException . . . . .	4031
6.850.2.2 UnsupportedOperationException . . . . .	4031
6.850.2.3 UnsupportedOperationException . . . . .	4031
6.850.2.4 UnsupportedOperationException . . . . .	4031
6.850.2.5 ~UnsupportedOperationException . . . . .	4031
6.851 decaf::net::URI Class Reference . . . . .	4031
6.851.1 Detailed Description . . . . .	4034
6.851.2 Constructor & Destructor Documentation . . . . .	4034
6.851.2.1 URI . . . . .	4034
6.851.2.2 URI . . . . .	4034
6.851.2.3 URI . . . . .	4034
6.851.2.4 URI . . . . .	4035
6.851.2.5 URI . . . . .	4035
6.851.2.6 URI . . . . .	4035
6.851.2.7 URI . . . . .	4036
6.851.2.8 ~URI . . . . .	4036
6.851.3 Member Function Documentation . . . . .	4036
6.851.3.1 compareTo . . . . .	4036
6.851.3.2 create . . . . .	4036
6.851.3.3 equals . . . . .	4037
6.851.3.4 getAuthority . . . . .	4037
6.851.3.5 getFragment . . . . .	4037
6.851.3.6 getHost . . . . .	4037
6.851.3.7 getPath . . . . .	4037
6.851.3.8 getPort . . . . .	4037
6.851.3.9 getQuery . . . . .	4037
6.851.3.10 getRawAuthority . . . . .	4038
6.851.3.11 getRawFragment . . . . .	4038
6.851.3.12 getRawPath . . . . .	4038
6.851.3.13 getRawQuery . . . . .	4038
6.851.3.14 getRawSchemeSpecificPart . . . . .	4039
6.851.3.15 getRawUserInfo . . . . .	4039
6.851.3.16 getScheme . . . . .	4039
6.851.3.17 getSchemeSpecificPart . . . . .	4039
6.851.3.18 getUserInfo . . . . .	4039
6.851.3.19 isAbsolute . . . . .	4040
6.851.3.20 isOpaque . . . . .	4040
6.851.3.21 normalize . . . . .	4040
6.851.3.22 operator< . . . . .	4040
6.851.3.23 operator== . . . . .	4041
6.851.3.24 parseServerAuthority . . . . .	4041
6.851.3.25 relativize . . . . .	4042
6.851.3.26 resolve . . . . .	4042
6.851.3.27 resolve . . . . .	4042
6.851.3.28 toString . . . . .	4043
6.851.3.29 toURL . . . . .	4043
6.852 decaf::internal::net::URIEncoderDecoder Class Reference . . . . .	4044
6.852.1 Constructor & Destructor Documentation . . . . .	4045

6.852.1.1 URLEncoderDecoder . . . . .	4045
6.852.1.2 ~URLEncoderDecoder . . . . .	4045
6.852.2 Member Function Documentation . . . . .	4045
6.852.2.1 decode . . . . .	4045
6.852.2.2 encodeOthers . . . . .	4045
6.852.2.3 quoteIllegal . . . . .	4046
6.852.2.4 validate . . . . .	4046
6.852.2.5 validateSimple . . . . .	4046
6.853decaf::internal::net::URIHelper Class Reference . . . . .	4047
6.853.1 Detailed Description . . . . .	4048
6.853.2 Constructor & Destructor Documentation . . . . .	4048
6.853.2.1 URIHelper . . . . .	4048
6.853.2.2 URIHelper . . . . .	4049
6.853.2.3 ~URIHelper . . . . .	4049
6.853.3 Member Function Documentation . . . . .	4049
6.853.3.1 isValidDomainName . . . . .	4049
6.853.3.2 isValidHexChar . . . . .	4049
6.853.3.3 isValidHost . . . . .	4049
6.853.3.4 isValidIP4Word . . . . .	4050
6.853.3.5 isValidIP6Address . . . . .	4050
6.853.3.6 isValidIPv4Address . . . . .	4050
6.853.3.7 parseAuthority . . . . .	4051
6.853.3.8 parseURI . . . . .	4051
6.853.3.9 validateAuthority . . . . .	4052
6.853.3.10validateFragment . . . . .	4052
6.853.3.11validatePath . . . . .	4052
6.853.3.12validateQuery . . . . .	4053
6.853.3.13validateScheme . . . . .	4053
6.853.3.14validateSsp . . . . .	4053
6.853.3.15validateUserInfo . . . . .	4054
6.854activemq::transport::failover::URIPool Class Reference . . . . .	4054
6.854.1 Constructor & Destructor Documentation . . . . .	4055
6.854.1.1 URIPool . . . . .	4055
6.854.1.2 URIPool . . . . .	4055
6.854.1.3 ~URIPool . . . . .	4055
6.854.2 Member Function Documentation . . . . .	4055
6.854.2.1 addURI . . . . .	4055
6.854.2.2 addURIs . . . . .	4056
6.854.2.3 getURI . . . . .	4056
6.854.2.4 isRandomize . . . . .	4056
6.854.2.5 removeURI . . . . .	4056
6.854.2.6 setRandomize . . . . .	4057
6.855activemq::util::URISupport Class Reference . . . . .	4057
6.855.1 Member Function Documentation . . . . .	4058
6.855.1.1 createQueryString . . . . .	4058
6.855.1.2 parseComposite . . . . .	4058
6.855.1.3 parseQuery . . . . .	4058
6.855.1.4 parseQuery . . . . .	4059
6.855.1.5 parseURL . . . . .	4059
6.856decaf::net::URISyntaxException Class Reference . . . . .	4060

6.856.1 Constructor & Destructor Documentation . . . . .	4061
6.856.1.1 URISyntaxException . . . . .	4061
6.856.1.2 URISyntaxException . . . . .	4061
6.856.1.3 URISyntaxException . . . . .	4061
6.856.1.4 URISyntaxException . . . . .	4061
6.856.1.5 URISyntaxException . . . . .	4061
6.856.1.6 URISyntaxException . . . . .	4062
6.856.1.7 URISyntaxException . . . . .	4062
6.856.1.8 URISyntaxException . . . . .	4062
6.856.1.9 ~URISyntaxException . . . . .	4063
6.856.2 Member Function Documentation . . . . .	4063
6.856.2.1 clone . . . . .	4063
6.856.2.2 getIndex . . . . .	4063
6.856.2.3 getInput . . . . .	4063
6.856.2.4 getReason . . . . .	4063
6.857decaf::internal::net::URIType Class Reference . . . . .	4063
6.857.1 Detailed Description . . . . .	4066
6.857.2 Constructor & Destructor Documentation . . . . .	4066
6.857.2.1 URIType . . . . .	4066
6.857.2.2 URIType . . . . .	4066
6.857.2.3 ~URIType . . . . .	4066
6.857.3 Member Function Documentation . . . . .	4066
6.857.3.1 getAuthority . . . . .	4066
6.857.3.2 getFragment . . . . .	4066
6.857.3.3 getHost . . . . .	4066
6.857.3.4 getPath . . . . .	4066
6.857.3.5 getPort . . . . .	4067
6.857.3.6 getQuery . . . . .	4067
6.857.3.7 getScheme . . . . .	4067
6.857.3.8 getSchemeSpecificPart . . . . .	4067
6.857.3.9 getSource . . . . .	4067
6.857.3.10getUserInfo . . . . .	4068
6.857.3.11isAbsolute . . . . .	4068
6.857.3.12isOpaque . . . . .	4068
6.857.3.13isServerAuthority . . . . .	4068
6.857.3.14isValid . . . . .	4068
6.857.3.15setAbsolute . . . . .	4068
6.857.3.16setAuthority . . . . .	4069
6.857.3.17setFragment . . . . .	4069
6.857.3.18setHost . . . . .	4069
6.857.3.19setOpaque . . . . .	4069
6.857.3.20setPath . . . . .	4069
6.857.3.21setPort . . . . .	4070
6.857.3.22setQuery . . . . .	4070
6.857.3.23setScheme . . . . .	4070
6.857.3.24setSchemeSpecificPart . . . . .	4070
6.857.3.25setServerAuthority . . . . .	4071
6.857.3.26setSource . . . . .	4071
6.857.3.27setUserInfo . . . . .	4071
6.857.3.28setValid . . . . .	4071

6.858	decaf::net::URL Class Reference . . . . .	4072
6.858.1	Detailed Description . . . . .	4072
6.858.2	Constructor & Destructor Documentation . . . . .	4073
6.858.2.1	URL . . . . .	4073
6.858.2.2	URL . . . . .	4073
6.858.2.3	~URL . . . . .	4073
6.859	decaf::net::URLDecoder Class Reference . . . . .	4074
6.859.1	Constructor & Destructor Documentation . . . . .	4074
6.859.1.1	~URLDecoder . . . . .	4074
6.859.2	Member Function Documentation . . . . .	4074
6.859.2.1	decode . . . . .	4074
6.860	decaf::net::URLEncoder Class Reference . . . . .	4074
6.860.1	Constructor & Destructor Documentation . . . . .	4075
6.860.1.1	~URLEncoder . . . . .	4075
6.860.2	Member Function Documentation . . . . .	4075
6.860.2.1	encode . . . . .	4075
6.861	activemq::util::Usage Class Reference . . . . .	4075
6.861.1	Constructor & Destructor Documentation . . . . .	4076
6.861.1.1	~Usage . . . . .	4076
6.861.2	Member Function Documentation . . . . .	4076
6.861.2.1	decreaseUsage . . . . .	4076
6.861.2.2	enqueueUsage . . . . .	4076
6.861.2.3	increaseUsage . . . . .	4077
6.861.2.4	isFull . . . . .	4077
6.861.2.5	waitForSpace . . . . .	4077
6.861.2.6	waitForSpace . . . . .	4077
6.862	decaf::io::UTFDataFormatException Class Reference . . . . .	4078
6.862.1	Detailed Description . . . . .	4078
6.862.2	Constructor & Destructor Documentation . . . . .	4079
6.862.2.1	UTFDataFormatException . . . . .	4079
6.862.2.2	UTFDataFormatException . . . . .	4079
6.862.2.3	UTFDataFormatException . . . . .	4079
6.862.2.4	UTFDataFormatException . . . . .	4079
6.862.2.5	UTFDataFormatException . . . . .	4079
6.862.2.6	UTFDataFormatException . . . . .	4080
6.862.2.7	~UTFDataFormatException . . . . .	4080
6.862.3	Member Function Documentation . . . . .	4080
6.862.3.1	clone . . . . .	4080
6.863	decaf::util::UUID Class Reference . . . . .	4080
6.863.1	Detailed Description . . . . .	4082
6.863.2	Constructor & Destructor Documentation . . . . .	4083
6.863.2.1	UUID . . . . .	4083
6.863.2.2	~UUID . . . . .	4083
6.863.3	Member Function Documentation . . . . .	4083
6.863.3.1	clockSequence . . . . .	4083
6.863.3.2	compareTo . . . . .	4083
6.863.3.3	equals . . . . .	4084
6.863.3.4	fromString . . . . .	4084
6.863.3.5	getLeastSignificantBits . . . . .	4084
6.863.3.6	getMostSignificantBits . . . . .	4084



6.863.3.7 nameUUIDFromBytes . . . . .	4084
6.863.3.8 nameUUIDFromBytes . . . . .	4085
6.863.3.9 node . . . . .	4085
6.863.3.10operator< . . . . .	4085
6.863.3.11operator== . . . . .	4086
6.863.3.12randomUUID . . . . .	4086
6.863.3.13timestamp . . . . .	4086
6.863.3.14toString . . . . .	4087
6.863.3.15variant . . . . .	4087
6.863.3.16version . . . . .	4087
6.864activemq::wireformat::WireFormat Class Reference . . . . .	4088
6.864.1 Detailed Description . . . . .	4089
6.864.2 Constructor & Destructor Documentation . . . . .	4089
6.864.2.1 ~WireFormat . . . . .	4089
6.864.3 Member Function Documentation . . . . .	4089
6.864.3.1 createNegotiator . . . . .	4089
6.864.3.2 getVersion . . . . .	4090
6.864.3.3 hasNegotiator . . . . .	4090
6.864.3.4 inReceive . . . . .	4090
6.864.3.5 marshal . . . . .	4091
6.864.3.6 setVersion . . . . .	4091
6.864.3.7 unmarshal . . . . .	4091
6.865activemq::wireformat::WireFormatFactory Class Reference . . . . .	4092
6.865.1 Detailed Description . . . . .	4092
6.865.2 Constructor & Destructor Documentation . . . . .	4093
6.865.2.1 ~WireFormatFactory . . . . .	4093
6.865.3 Member Function Documentation . . . . .	4093
6.865.3.1 createWireFormat . . . . .	4093
6.866activemq::commands::WireFormatInfo Class Reference . . . . .	4093
6.866.1 Constructor & Destructor Documentation . . . . .	4096
6.866.1.1 WireFormatInfo . . . . .	4096
6.866.1.2 ~WireFormatInfo . . . . .	4096
6.866.2 Member Function Documentation . . . . .	4096
6.866.2.1 afterUnmarshal . . . . .	4096
6.866.2.2 beforeMarshal . . . . .	4097
6.866.2.3 cloneDataStructure . . . . .	4097
6.866.2.4 copyDataStructure . . . . .	4097
6.866.2.5 equals . . . . .	4097
6.866.2.6 getCacheSize . . . . .	4098
6.866.2.7 getDataStructureType . . . . .	4098
6.866.2.8 getMagic . . . . .	4098
6.866.2.9 getMarshaledProperties . . . . .	4098
6.866.2.10getMaxInactivityDuration . . . . .	4098
6.866.2.11getMaxInactivityDurationInitialDelay . . . . .	4099
6.866.2.12getProperties . . . . .	4099
6.866.2.13getProperties . . . . .	4099
6.866.2.14getVersion . . . . .	4099
6.866.2.15isCacheEnabled . . . . .	4099
6.866.2.16isMarshalAware . . . . .	4100
6.866.2.17isSizePrefixDisabled . . . . .	4100

6.866.2.18	isStackTraceEnabled	4100
6.866.2.19	isTcpNoDelayEnabled	4100
6.866.2.20	isTightEncodingEnabled	4100
6.866.2.21	isValid	4101
6.866.2.22	isWireFormatInfo	4101
6.866.2.23	setCacheEnabled	4101
6.866.2.24	setCacheSize	4101
6.866.2.25	setMagic	4101
6.866.2.26	setMarshaledProperties	4102
6.866.2.27	setMaxInactivityDuration	4102
6.866.2.28	setMaxInactivityDurationInitialDelay	4102
6.866.2.29	setProperties	4102
6.866.2.30	setSizePrefixDisabled	4102
6.866.2.31	setStackTraceEnabled	4103
6.866.2.32	setTcpNoDelayEnabled	4103
6.866.2.33	setTightEncodingEnabled	4103
6.866.2.34	setVersion	4103
6.866.2.35	toString	4104
6.866.2.36	visit	4104
6.866.3	Field Documentation	4104
6.866.3.1	ID_WIREFORMATINFO	4104
6.867	activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller	
	Class Reference	4104
6.867.1	Detailed Description	4105
6.867.2	Constructor & Destructor Documentation	4106
6.867.2.1	WireFormatInfoMarshaller	4106
6.867.2.2	~WireFormatInfoMarshaller	4106
6.867.3	Member Function Documentation	4106
6.867.3.1	createObject	4106
6.867.3.2	getDataStructureType	4106
6.867.3.3	looseMarshal	4106
6.867.3.4	looseUnmarshal	4107
6.867.3.5	tightMarshal1	4107
6.867.3.6	tightMarshal2	4108
6.867.3.7	tightUnmarshal	4108
6.868	activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller	
	Class Reference	4109
6.868.1	Detailed Description	4110
6.868.2	Constructor & Destructor Documentation	4110
6.868.2.1	WireFormatInfoMarshaller	4110
6.868.2.2	~WireFormatInfoMarshaller	4110
6.868.3	Member Function Documentation	4110
6.868.3.1	createObject	4110
6.868.3.2	getDataStructureType	4110
6.868.3.3	looseMarshal	4110
6.868.3.4	looseUnmarshal	4111
6.868.3.5	tightMarshal1	4111
6.868.3.6	tightMarshal2	4112
6.868.3.7	tightUnmarshal	4112

6.869	activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller	
	Class Reference . . . . .	4113
6.869.1	Detailed Description . . . . .	4114
6.869.2	Constructor & Destructor Documentation . . . . .	4114
	6.869.2.1 WireFormatInfoMarshaller . . . . .	4114
	6.869.2.2 ~WireFormatInfoMarshaller . . . . .	4114
6.869.3	Member Function Documentation . . . . .	4114
	6.869.3.1 createObject . . . . .	4114
	6.869.3.2 getDataStructureType . . . . .	4114
	6.869.3.3 looseMarshal . . . . .	4114
	6.869.3.4 looseUnmarshal . . . . .	4115
	6.869.3.5 tightMarshal1 . . . . .	4115
	6.869.3.6 tightMarshal2 . . . . .	4116
	6.869.3.7 tightUnmarshal . . . . .	4116
6.870	activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller	
	Class Reference . . . . .	4117
6.870.1	Detailed Description . . . . .	4118
6.870.2	Constructor & Destructor Documentation . . . . .	4118
	6.870.2.1 WireFormatInfoMarshaller . . . . .	4118
	6.870.2.2 ~WireFormatInfoMarshaller . . . . .	4118
6.870.3	Member Function Documentation . . . . .	4118
	6.870.3.1 createObject . . . . .	4118
	6.870.3.2 getDataStructureType . . . . .	4118
	6.870.3.3 looseMarshal . . . . .	4118
	6.870.3.4 looseUnmarshal . . . . .	4119
	6.870.3.5 tightMarshal1 . . . . .	4119
	6.870.3.6 tightMarshal2 . . . . .	4120
	6.870.3.7 tightUnmarshal . . . . .	4120
6.871	activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller	
	Class Reference . . . . .	4121
6.871.1	Detailed Description . . . . .	4122
6.871.2	Constructor & Destructor Documentation . . . . .	4122
	6.871.2.1 WireFormatInfoMarshaller . . . . .	4122
	6.871.2.2 ~WireFormatInfoMarshaller . . . . .	4122
6.871.3	Member Function Documentation . . . . .	4122
	6.871.3.1 createObject . . . . .	4122
	6.871.3.2 getDataStructureType . . . . .	4122
	6.871.3.3 looseMarshal . . . . .	4122
	6.871.3.4 looseUnmarshal . . . . .	4123
	6.871.3.5 tightMarshal1 . . . . .	4123
	6.871.3.6 tightMarshal2 . . . . .	4124
	6.871.3.7 tightUnmarshal . . . . .	4124
6.872	activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller	
	Class Reference . . . . .	4125
6.872.1	Detailed Description . . . . .	4126
6.872.2	Constructor & Destructor Documentation . . . . .	4126
	6.872.2.1 WireFormatInfoMarshaller . . . . .	4126
	6.872.2.2 ~WireFormatInfoMarshaller . . . . .	4126
6.872.3	Member Function Documentation . . . . .	4126
	6.872.3.1 createObject . . . . .	4126

6.872.3.2	getDataStructureType . . . . .	4126
6.872.3.3	looseMarshal . . . . .	4126
6.872.3.4	looseUnmarshal . . . . .	4127
6.872.3.5	tightMarshal1 . . . . .	4127
6.872.3.6	tightMarshal2 . . . . .	4128
6.872.3.7	tightUnmarshal . . . . .	4128
6.873	activemq::wireformat::WireFormatNegotiator Class Reference . . . . .	4129
6.873.1	Detailed Description . . . . .	4129
6.873.2	Constructor & Destructor Documentation . . . . .	4129
6.873.2.1	WireFormatNegotiator . . . . .	4129
6.873.2.2	~WireFormatNegotiator . . . . .	4129
6.874	activemq::wireformat::WireFormatRegistry Class Reference . . . . .	4129
6.874.1	Detailed Description . . . . .	4130
6.874.2	Constructor & Destructor Documentation . . . . .	4131
6.874.2.1	~WireFormatRegistry . . . . .	4131
6.874.3	Member Function Documentation . . . . .	4131
6.874.3.1	findFactory . . . . .	4131
6.874.3.2	getInstance . . . . .	4131
6.874.3.3	getWireFormatNames . . . . .	4131
6.874.3.4	registerFactory . . . . .	4132
6.874.3.5	unregisterFactory . . . . .	4132
6.875	activemq::transport::inactivity::WriteChecker Class Reference . . . . .	4132
6.875.1	Detailed Description . . . . .	4133
6.875.2	Constructor & Destructor Documentation . . . . .	4133
6.875.2.1	WriteChecker . . . . .	4133
6.875.2.2	~WriteChecker . . . . .	4133
6.875.3	Member Function Documentation . . . . .	4133
6.875.3.1	run . . . . .	4133
6.876	decaf::io::Writer Class Reference . . . . .	4133
6.876.1	Constructor & Destructor Documentation . . . . .	4135
6.876.1.1	Writer . . . . .	4135
6.876.1.2	~Writer . . . . .	4135
6.876.2	Member Function Documentation . . . . .	4135
6.876.2.1	append . . . . .	4135
6.876.2.2	append . . . . .	4136
6.876.2.3	append . . . . .	4136
6.876.2.4	doAppendChar . . . . .	4137
6.876.2.5	doAppendCharSequence . . . . .	4137
6.876.2.6	doAppendCharSequenceStartEnd . . . . .	4137
6.876.2.7	doWriteArray . . . . .	4137
6.876.2.8	doWriteArrayBounded . . . . .	4137
6.876.2.9	doWriteChar . . . . .	4137
6.876.2.10	doWriteString . . . . .	4137
6.876.2.11	doWriteStringBounded . . . . .	4137
6.876.2.12	doWriteVector . . . . .	4137
6.876.2.13	write . . . . .	4137
6.876.2.14	write . . . . .	4138
6.876.2.15	write . . . . .	4138
6.876.2.16	write . . . . .	4138
6.876.2.17	write . . . . .	4139

6.876.2.18write . . . . .	4139
6.877decaf::security::auth::x500::X500Principal Class Reference . . . . .	4140
6.877.1 Constructor & Destructor Documentation . . . . .	4140
6.877.1.1 ~X500Principal . . . . .	4140
6.877.2 Member Function Documentation . . . . .	4140
6.877.2.1 getEncoded . . . . .	4140
6.877.2.2 getName . . . . .	4140
6.877.2.3 hashCode . . . . .	4140
6.878decaf::security::cert::X509Certificate Class Reference . . . . .	4141
6.878.1 Detailed Description . . . . .	4141
6.878.2 Constructor & Destructor Documentation . . . . .	4142
6.878.2.1 ~X509Certificate . . . . .	4142
6.878.3 Member Function Documentation . . . . .	4142
6.878.3.1 checkValidity . . . . .	4142
6.878.3.2 checkValidity . . . . .	4142
6.878.3.3 getBasicConstraints . . . . .	4142
6.878.3.4 getIssuerUniqueID . . . . .	4142
6.878.3.5 getIssuerX500Principal . . . . .	4142
6.878.3.6 getKeyUsage . . . . .	4142
6.878.3.7 getNotAfter . . . . .	4142
6.878.3.8 getNotBefore . . . . .	4142
6.878.3.9 getSigAlgName . . . . .	4142
6.878.3.10getSigAlgOID . . . . .	4142
6.878.3.11getSigAlgParams . . . . .	4142
6.878.3.12getSignature . . . . .	4142
6.878.3.13getSubjectUniqueID . . . . .	4142
6.878.3.14getSubjectX500Principal . . . . .	4142
6.878.3.15getTBSCertificate . . . . .	4142
6.878.3.16getVersion . . . . .	4142
6.879activemq::commands::XATransactionId Class Reference . . . . .	4143
6.879.1 Member Typedef Documentation . . . . .	4144
6.879.1.1 COMPARATOR . . . . .	4144
6.879.2 Constructor & Destructor Documentation . . . . .	4144
6.879.2.1 XATransactionId . . . . .	4144
6.879.2.2 XATransactionId . . . . .	4144
6.879.2.3 ~XATransactionId . . . . .	4144
6.879.3 Member Function Documentation . . . . .	4144
6.879.3.1 cloneDataStructure . . . . .	4144
6.879.3.2 compareTo . . . . .	4145
6.879.3.3 copyDataStructure . . . . .	4145
6.879.3.4 equals . . . . .	4145
6.879.3.5 equals . . . . .	4145
6.879.3.6 getBranchQualifier . . . . .	4145
6.879.3.7 getBranchQualifier . . . . .	4145
6.879.3.8 getDataStructureType . . . . .	4145
6.879.3.9 getFormatId . . . . .	4146
6.879.3.10getGlobalTransactionId . . . . .	4146
6.879.3.11getGlobalTransactionId . . . . .	4146
6.879.3.12operator< . . . . .	4146
6.879.3.13operator= . . . . .	4146

6.879.3.14	operator==	4146
6.879.3.15	setBranchQualifier	4146
6.879.3.16	setFormatId	4146
6.879.3.17	setGlobalTransactionId	4146
6.879.3.18	toString	4146
6.879.4	Field Documentation	4147
6.879.4.1	branchQualifier	4147
6.879.4.2	formatId	4147
6.879.4.3	globalTransactionId	4147
6.879.4.4	ID_XATRANSACTIONID	4147
6.880	activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller	
	Class Reference	4147
6.880.1	Detailed Description	4148
6.880.2	Constructor & Destructor Documentation	4148
6.880.2.1	XATransactionIdMarshaller	4148
6.880.2.2	~XATransactionIdMarshaller	4148
6.880.3	Member Function Documentation	4148
6.880.3.1	createObject	4148
6.880.3.2	getDataStructureType	4149
6.880.3.3	looseMarshal	4149
6.880.3.4	looseUnmarshal	4149
6.880.3.5	tightMarshal1	4150
6.880.3.6	tightMarshal2	4150
6.880.3.7	tightUnmarshal	4151
6.881	activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller	
	Class Reference	4151
6.881.1	Detailed Description	4152
6.881.2	Constructor & Destructor Documentation	4152
6.881.2.1	XATransactionIdMarshaller	4152
6.881.2.2	~XATransactionIdMarshaller	4152
6.881.3	Member Function Documentation	4152
6.881.3.1	createObject	4152
6.881.3.2	getDataStructureType	4153
6.881.3.3	looseMarshal	4153
6.881.3.4	looseUnmarshal	4153
6.881.3.5	tightMarshal1	4154
6.881.3.6	tightMarshal2	4154
6.881.3.7	tightUnmarshal	4155
6.882	activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller	
	Class Reference	4155
6.882.1	Detailed Description	4156
6.882.2	Constructor & Destructor Documentation	4157
6.882.2.1	XATransactionIdMarshaller	4157
6.882.2.2	~XATransactionIdMarshaller	4157
6.882.3	Member Function Documentation	4157
6.882.3.1	createObject	4157
6.882.3.2	getDataStructureType	4157
6.882.3.3	looseMarshal	4157
6.882.3.4	looseUnmarshal	4158
6.882.3.5	tightMarshal1	4158

6.882.3.6	tightMarshal2	4159
6.882.3.7	tightUnmarshal	4159
6.883	activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller	
	Class Reference	4160
6.883.1	Detailed Description	4161
6.883.2	Constructor & Destructor Documentation	4161
6.883.2.1	XATransactionIdMarshaller	4161
6.883.2.2	~XATransactionIdMarshaller	4161
6.883.3	Member Function Documentation	4161
6.883.3.1	createObject	4161
6.883.3.2	getDataStructureType	4161
6.883.3.3	looseMarshal	4161
6.883.3.4	looseUnmarshal	4162
6.883.3.5	tightMarshal1	4162
6.883.3.6	tightMarshal2	4163
6.883.3.7	tightUnmarshal	4163
6.884	activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller	
	Class Reference	4164
6.884.1	Detailed Description	4165
6.884.2	Constructor & Destructor Documentation	4165
6.884.2.1	XATransactionIdMarshaller	4165
6.884.2.2	~XATransactionIdMarshaller	4165
6.884.3	Member Function Documentation	4165
6.884.3.1	createObject	4165
6.884.3.2	getDataStructureType	4165
6.884.3.3	looseMarshal	4166
6.884.3.4	looseUnmarshal	4166
6.884.3.5	tightMarshal1	4167
6.884.3.6	tightMarshal2	4167
6.884.3.7	tightUnmarshal	4168
6.885	activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller	
	Class Reference	4168
6.885.1	Detailed Description	4169
6.885.2	Constructor & Destructor Documentation	4169
6.885.2.1	XATransactionIdMarshaller	4169
6.885.2.2	~XATransactionIdMarshaller	4169
6.885.3	Member Function Documentation	4169
6.885.3.1	createObject	4169
6.885.3.2	getDataStructureType	4170
6.885.3.3	looseMarshal	4170
6.885.3.4	looseUnmarshal	4170
6.885.3.5	tightMarshal1	4171
6.885.3.6	tightMarshal2	4171
6.885.3.7	tightUnmarshal	4172
6.886	decaf::util::logging::XMLFormatter Class Reference	4172
6.886.1	Detailed Description	4173
6.886.2	Constructor & Destructor Documentation	4173
6.886.2.1	XMLFormatter	4173
6.886.2.2	~XMLFormatter	4173
6.886.3	Member Function Documentation	4173

6.886.3.1	format	4173
6.886.3.2	getHead	4174
6.886.3.3	getTail	4174
6.887	z_stream_s Struct Reference	4174
6.887.1	Field Documentation	4175
6.887.1.1	adler	4175
6.887.1.2	avail_in	4175
6.887.1.3	avail_out	4175
6.887.1.4	data_type	4175
6.887.1.5	msg	4175
6.887.1.6	next_in	4175
6.887.1.7	next_out	4175
6.887.1.8	opaque	4175
6.887.1.9	reserved	4175
6.887.1.10	state	4175
6.887.1.11	total_in	4175
6.887.1.12	total_out	4175
6.887.1.13	zalloc	4175
6.887.1.14	zfree	4175
6.888	decaf::util::zip::ZipException Class Reference	4175
6.888.1	Constructor & Destructor Documentation	4176
6.888.1.1	ZipException	4176
6.888.1.2	ZipException	4176
6.888.1.3	ZipException	4177
6.888.1.4	ZipException	4177
6.888.1.5	ZipException	4177
6.888.1.6	ZipException	4177
6.888.1.7	~ZipException	4178
6.888.2	Member Function Documentation	4178
6.888.2.1	clone	4178
<b>7</b>	<b>File Documentation</b>	<b>4179</b>
7.1	src/main/activemq/cmsutil/CachedConsumer.h File Reference	4179
7.2	src/main/activemq/cmsutil/CachedProducer.h File Reference	4179
7.3	src/main/activemq/cmsutil/CmsAccessor.h File Reference	4180
7.4	src/main/activemq/cmsutil/CmsDestinationAccessor.h File Reference	4180
7.5	src/main/activemq/cmsutil/CmsTemplate.h File Reference	4181
7.6	src/main/activemq/cmsutil/DestinationResolver.h File Reference	4182
7.7	src/main/activemq/cmsutil/DynamicDestinationResolver.h File Reference	4182
7.8	src/main/activemq/cmsutil/MessageCreator.h File Reference	4183
7.9	src/main/activemq/cmsutil/PooledSession.h File Reference	4183
7.10	src/main/activemq/cmsutil/ProducerCallback.h File Reference	4184
7.11	src/main/activemq/cmsutil/ResourceLifecycleManager.h File Reference	4184
7.12	src/main/decaf/internal/util/ResourceLifecycleManager.h File Reference	4185
7.13	src/main/activemq/cmsutil/SessionCallback.h File Reference	4185
7.14	src/main/activemq/cmsutil/SessionPool.h File Reference	4186
7.15	src/main/activemq/commands/ActiveMQBlobMessage.h File Reference	4186
7.16	src/main/activemq/commands/ActiveMQBytesMessage.h File Reference	4187
7.17	src/main/activemq/commands/ActiveMQDestination.h File Reference	4188



7.18	src/main/activemq/commands/ActiveMQMapMessage.h File Reference	4188
7.19	src/main/activemq/commands/ActiveMQMessage.h File Reference . .	4189
7.20	src/main/activemq/commands/ActiveMQMessageTemplate.h File Reference . . . . .	4189
7.21	src/main/activemq/commands/ActiveMQObjectMessage.h File Reference . . . . .	4190
7.22	src/main/activemq/commands/ActiveMQQueue.h File Reference . . .	4191
7.23	src/main/activemq/commands/ActiveMQStreamMessage.h File Reference . . . . .	4191
7.24	src/main/activemq/commands/ActiveMQTempDestination.h File Reference . . . . .	4192
7.25	src/main/activemq/commands/ActiveMQTempQueue.h File Reference	4193
7.26	src/main/activemq/commands/ActiveMQTempTopic.h File Reference	4193
7.27	src/main/activemq/commands/ActiveMQTextMessage.h File Reference	4194
7.28	src/main/activemq/commands/ActiveMQTopic.h File Reference . . .	4194
7.29	src/main/activemq/commands/BaseCommand.h File Reference . . . .	4195
7.30	src/main/activemq/commands/BaseDataStructure.h File Reference . .	4195
7.31	src/main/activemq/commands/BooleanExpression.h File Reference . .	4196
7.32	src/main/activemq/commands/BrokerError.h File Reference . . . . .	4196
7.33	src/main/activemq/commands/BrokerId.h File Reference . . . . .	4197
7.34	src/main/activemq/commands/BrokerInfo.h File Reference . . . . .	4197
7.35	src/main/activemq/commands/Command.h File Reference . . . . .	4198
7.36	src/main/activemq/commands/ConnectionControl.h File Reference . .	4199
7.37	src/main/activemq/commands/ConnectionError.h File Reference . . .	4199
7.38	src/main/activemq/commands/ConnectionId.h File Reference . . . . .	4200
7.39	src/main/activemq/commands/ConnectionInfo.h File Reference . . . .	4200
7.40	src/main/activemq/commands/ConsumerControl.h File Reference . . .	4201
7.41	src/main/activemq/commands/ConsumerId.h File Reference . . . . .	4201
7.42	src/main/activemq/commands/ConsumerInfo.h File Reference . . . . .	4202
7.43	src/main/activemq/commands/ControlCommand.h File Reference . . .	4203
7.44	src/main/activemq/commands/DataArrayResponse.h File Reference . .	4203
7.45	src/main/activemq/commands/DataResponse.h File Reference . . . . .	4204
7.46	src/main/activemq/commands/DataStructure.h File Reference . . . . .	4204
7.47	src/main/activemq/commands/DestinationInfo.h File Reference . . . .	4205
7.48	src/main/activemq/commands/DiscoveryEvent.h File Reference . . . .	4205
7.49	src/main/activemq/commands/ExceptionResponse.h File Reference . .	4206
7.50	src/main/activemq/commands/FlushCommand.h File Reference . . . .	4206
7.51	src/main/activemq/commands/IntegerResponse.h File Reference . . . .	4207
7.52	src/main/activemq/commands/JournalQueueAck.h File Reference . . .	4207
7.53	src/main/activemq/commands/JournalTopicAck.h File Reference . . . .	4208
7.54	src/main/activemq/commands/JournalTrace.h File Reference . . . . .	4209
7.55	src/main/activemq/commands/JournalTransaction.h File Reference . .	4209
7.56	src/main/activemq/commands/KeepAliveInfo.h File Reference . . . . .	4210
7.57	src/main/activemq/commands/LastPartialCommand.h File Reference . .	4210
7.58	src/main/activemq/commands/LocalTransactionId.h File Reference . .	4211
7.59	src/main/activemq/commands/Message.h File Reference . . . . .	4211
7.60	src/main/cms/Message.h File Reference . . . . .	4212
7.61	src/main/activemq/commands/MessageAck.h File Reference . . . . .	4213
7.62	src/main/activemq/commands/MessageDispatch.h File Reference . . .	4213

7.63	src/main/activemq/commands/MessageDispatchNotification.h File Reference . . . . .	4214
7.64	src/main/activemq/commands/MessageId.h File Reference . . . . .	4215
7.65	src/main/activemq/commands/MessagePull.h File Reference . . . . .	4215
7.66	src/main/activemq/commands/NetworkBridgeFilter.h File Reference . . . . .	4216
7.67	src/main/activemq/commands/PartialCommand.h File Reference . . . . .	4216
7.68	src/main/activemq/commands/ProducerAck.h File Reference . . . . .	4217
7.69	src/main/activemq/commands/ProducerId.h File Reference . . . . .	4217
7.70	src/main/activemq/commands/ProducerInfo.h File Reference . . . . .	4218
7.71	src/main/activemq/commands/RemoveInfo.h File Reference . . . . .	4219
7.72	src/main/activemq/commands/RemoveSubscriptionInfo.h File Reference . . . . .	4219
7.73	src/main/activemq/commands/ReplayCommand.h File Reference . . . . .	4220
7.74	src/main/activemq/commands/Response.h File Reference . . . . .	4220
7.75	src/main/activemq/commands/SessionId.h File Reference . . . . .	4221
7.76	src/main/activemq/commands/SessionInfo.h File Reference . . . . .	4221
7.77	src/main/activemq/commands/ShutdownInfo.h File Reference . . . . .	4222
7.78	src/main/activemq/commands/SubscriptionInfo.h File Reference . . . . .	4222
7.79	src/main/activemq/commands/TransactionId.h File Reference . . . . .	4223
7.80	src/main/activemq/commands/TransactionInfo.h File Reference . . . . .	4223
7.81	src/main/activemq/commands/WireFormatInfo.h File Reference . . . . .	4224
7.82	src/main/activemq/commands/XATransactionId.h File Reference . . . . .	4224
7.83	src/main/activemq/core/ActiveMQAckHandler.h File Reference . . . . .	4225
7.84	src/main/activemq/core/ActiveMQConnection.h File Reference . . . . .	4225
7.85	src/main/activemq/core/ActiveMQConnectionFactory.h File Reference . . . . .	4226
7.86	src/main/activemq/core/ActiveMQConnectionMetaData.h File Reference . . . . .	4227
7.87	src/main/activemq/core/ActiveMQConstants.h File Reference . . . . .	4227
7.88	src/main/activemq/core/ActiveMQConsumer.h File Reference . . . . .	4228
7.89	src/main/activemq/core/ActiveMQProducer.h File Reference . . . . .	4229
7.90	src/main/activemq/core/ActiveMQQueueBrowser.h File Reference . . . . .	4230
7.91	src/main/activemq/core/ActiveMQSession.h File Reference . . . . .	4230
7.92	src/main/activemq/core/ActiveMQSessionExecutor.h File Reference . . . . .	4231
7.93	src/main/activemq/core/ActiveMQTransactionContext.h File Reference . . . . .	4232
7.94	src/main/activemq/core/DispatchData.h File Reference . . . . .	4233
7.95	src/main/activemq/core/Dispatcher.h File Reference . . . . .	4233
7.96	src/main/activemq/core/MessageDispatchChannel.h File Reference . . . . .	4234
7.97	src/main/activemq/core/policies/DefaultPrefetchPolicy.h File Reference . . . . .	4234
7.98	src/main/activemq/core/policies/DefaultRedeliveryPolicy.h File Reference . . . . .	4235
7.99	src/main/activemq/core/PrefetchPolicy.h File Reference . . . . .	4235
7.100	src/main/activemq/core/RedeliveryPolicy.h File Reference . . . . .	4236
7.101	src/main/activemq/core/Synchronization.h File Reference . . . . .	4236
7.102	src/main/activemq/exceptions/ActiveMQException.h File Reference . . . . .	4237
7.103	src/main/activemq/exceptions/BrokerException.h File Reference . . . . .	4237
7.104	src/main/activemq/exceptions/ExceptionDefines.h File Reference . . . . .	4238
7.104.1	Define Documentation . . . . .	4238
7.104.1.1	AMQ_CATCH_EXCEPTION_CONVERT . . . . .	4238
7.104.1.2	AMQ_CATCH_NOTHROW . . . . .	4239
7.104.1.3	AMQ_CATCH_RETHROW . . . . .	4239
7.104.1.4	AMQ_CATCHALL_NOTHROW . . . . .	4239
7.104.1.5	AMQ_CATCHALL_THROW . . . . .	4240

7.105src/main/decaf/lang/exceptions/ExceptionDefines.h File Reference . . .	4240
7.105.1 Define Documentation . . . . .	4241
7.105.1.1 DECAF_CATCH_EXCEPTION_CONVERT . . . . .	4241
7.105.1.2 DECAF_CATCH_NOTHROW . . . . .	4241
7.105.1.3 DECAF_CATCH_RETHROW . . . . .	4241
7.105.1.4 DECAF_CATCHALL_NOTHROW . . . . .	4242
7.105.1.5 DECAF_CATCHALL_THROW . . . . .	4242
7.106src/main/activemq/io/LoggingInputStream.h File Reference . . . . .	4242
7.107src/main/activemq/io/LoggingOutputStream.h File Reference . . . . .	4243
7.108src/main/activemq/library/ActiveMQCPP.h File Reference . . . . .	4243
7.109src/main/activemq/state/CommandVisitor.h File Reference . . . . .	4244
7.110src/main/activemq/state/CommandVisitorAdapter.h File Reference . . .	4244
7.111src/main/activemq/state/ConnectionState.h File Reference . . . . .	4245
7.112src/main/activemq/state/ConnectionStateTracker.h File Reference . . .	4246
7.113src/main/activemq/state/ConsumerState.h File Reference . . . . .	4247
7.114src/main/activemq/state/ProducerState.h File Reference . . . . .	4248
7.115src/main/activemq/state/SessionState.h File Reference . . . . .	4248
7.116src/main/activemq/state/Tracked.h File Reference . . . . .	4249
7.117src/main/activemq/state/TransactionState.h File Reference . . . . .	4249
7.118src/main/activemq/threads/CompositeTask.h File Reference . . . . .	4250
7.119src/main/activemq/threads/CompositeTaskRunner.h File Reference . . .	4250
7.120src/main/activemq/threads/DedicatedTaskRunner.h File Reference . . .	4251
7.121src/main/activemq/threads/Task.h File Reference . . . . .	4252
7.122src/main/activemq/threads/TaskRunner.h File Reference . . . . .	4252
7.123src/main/activemq/transport/AbstractTransportFactory.h File Reference	4253
7.124src/main/activemq/transport/CompositeTransport.h File Reference . . .	4253
7.125src/main/activemq/transport/correlator/FutureResponse.h File Reference	4254
7.126src/main/activemq/transport/correlator/ResponseCorrelator.h File Ref- erence . . . . .	4254
7.127src/main/activemq/transport/DefaultTransportListener.h File Reference	4255
7.128src/main/activemq/transport/failover/BackupTransport.h File Reference	4256
7.129src/main/activemq/transport/failover/BackupTransportPool.h File Ref- erence . . . . .	4256
7.130src/main/activemq/transport/failover/CloseTransportsTask.h File Ref- erence . . . . .	4257
7.131src/main/activemq/transport/failover/FailoverTransport.h File Reference	4257
7.132src/main/activemq/transport/failover/FailoverTransportFactory.h File Ref- erence . . . . .	4258
7.133src/main/activemq/transport/failover/FailoverTransportListener.h File Ref- erence . . . . .	4259
7.134src/main/activemq/transport/failover/URIPool.h File Reference . . . .	4260
7.135src/main/activemq/transport/inactivity/InactivityMonitor.h File Refer- erence . . . . .	4260
7.136src/main/activemq/transport/inactivity/ReadChecker.h File Reference . .	4261
7.137src/main/activemq/transport/inactivity/WriteChecker.h File Reference	4261
7.138src/main/activemq/transport/IOTransport.h File Reference . . . . .	4262
7.139src/main/activemq/transport/logging/LoggingTransport.h File Reference	4263
7.140src/main/activemq/transport/mock/InternalCommandListener.h File Ref- erence . . . . .	4263
7.141src/main/activemq/transport/mock/MockTransport.h File Reference . .	4264

7.142src/main/activemq/transport/mock/MockTransportFactory.h File Reference . . . . .	4265
7.143src/main/activemq/transport/mock/ResponseBuilder.h File Reference . . . . .	4265
7.144src/main/activemq/transport/tcp/SslTransport.h File Reference . . . . .	4266
7.145src/main/activemq/transport/tcp/SslTransportFactory.h File Reference . . . . .	4266
7.146src/main/activemq/transport/tcp/TcpTransport.h File Reference . . . . .	4267
7.147src/main/activemq/transport/tcp/TcpTransportFactory.h File Reference . . . . .	4268
7.148src/main/activemq/transport/Transport.h File Reference . . . . .	4268
7.149src/main/activemq/transport/TransportFactory.h File Reference . . . . .	4269
7.150src/main/activemq/transport/TransportFilter.h File Reference . . . . .	4270
7.151src/main/activemq/transport/TransportListener.h File Reference . . . . .	4270
7.152src/main/activemq/transport/TransportRegistry.h File Reference . . . . .	4271
7.153src/main/activemq/util/ActiveMQProperties.h File Reference . . . . .	4271
7.154src/main/activemq/util/CMSExceptionSupport.h File Reference . . . . .	4272
7.154.1 Define Documentation . . . . .	4273
7.154.1.1 AMQ_CATCH_ALL_THROW_CMSEXCEPTION . . . . .	4273
7.155src/main/activemq/util/CompositeData.h File Reference . . . . .	4274
7.156src/main/activemq/util/Config.h File Reference . . . . .	4274
7.156.1 Define Documentation . . . . .	4275
7.156.1.1 AMQCPP_API . . . . .	4275
7.156.1.2 HAVE_PTHREAD_H . . . . .	4275
7.156.1.3 HAVE_UUID_T . . . . .	4275
7.156.1.4 HAVE_UUID_UUID_H . . . . .	4275
7.157src/main/cms/Config.h File Reference . . . . .	4275
7.157.1 Define Documentation . . . . .	4275
7.157.1.1 CMS_API . . . . .	4275
7.158src/main/decaf/util/Config.h File Reference . . . . .	4275
7.158.1 Define Documentation . . . . .	4275
7.158.1.1 DECAF_API . . . . .	4275
7.158.1.2 DECAF_UNUSED . . . . .	4275
7.158.1.3 HAVE_PTHREAD_H . . . . .	4275
7.158.1.4 HAVE_UUID_T . . . . .	4275
7.158.1.5 HAVE_UUID_UUID_H . . . . .	4275
7.159src/main/activemq/util/IdGenerator.h File Reference . . . . .	4275
7.160src/main/activemq/util/LongSequenceGenerator.h File Reference . . . . .	4276
7.161src/main/activemq/util/MarshallingSupport.h File Reference . . . . .	4276
7.162src/main/activemq/util/MemoryUsage.h File Reference . . . . .	4277
7.163src/main/activemq/util/PrimitiveList.h File Reference . . . . .	4277
7.164src/main/activemq/util/PrimitiveMap.h File Reference . . . . .	4278
7.165src/main/activemq/util/PrimitiveValueConverter.h File Reference . . . . .	4279
7.166src/main/activemq/util/PrimitiveValueNode.h File Reference . . . . .	4279
7.167src/main/activemq/util/URISupport.h File Reference . . . . .	4280
7.168src/main/activemq/util/Usage.h File Reference . . . . .	4280
7.169src/main/activemq/wireformat/MarshalAware.h File Reference . . . . .	4281
7.170src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h File Reference . . . . .	4281
7.171src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h File Reference . . . . .	4282
7.172src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h File Reference . . . . .	4282

7.173src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQBlobMessageMarshaller.h	
File Reference . . . . .	4283
7.174src/main/activemq/wireformat/openwire/marshall/v2/ActiveMQBlobMessageMarshaller.h	
File Reference . . . . .	4284
7.175src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQBlobMessageMarshaller.h	
File Reference . . . . .	4285
7.176src/main/activemq/wireformat/openwire/marshall/v4/ActiveMQBlobMessageMarshaller.h	
File Reference . . . . .	4285
7.177src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQBlobMessageMarshaller.h	
File Reference . . . . .	4286
7.178src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQBlobMessageMarshaller.h	
File Reference . . . . .	4287
7.179src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQBytesMessageMarshaller.h	
File Reference . . . . .	4288
7.180src/main/activemq/wireformat/openwire/marshall/v2/ActiveMQBytesMessageMarshaller.h	
File Reference . . . . .	4288
7.181src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQBytesMessageMarshaller.h	
File Reference . . . . .	4289
7.182src/main/activemq/wireformat/openwire/marshall/v4/ActiveMQBytesMessageMarshaller.h	
File Reference . . . . .	4290
7.183src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQBytesMessageMarshaller.h	
File Reference . . . . .	4291
7.184src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQBytesMessageMarshaller.h	
File Reference . . . . .	4291
7.185src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQDestinationMarshaller.h	
File Reference . . . . .	4292
7.186src/main/activemq/wireformat/openwire/marshall/v2/ActiveMQDestinationMarshaller.h	
File Reference . . . . .	4293
7.187src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQDestinationMarshaller.h	
File Reference . . . . .	4294
7.188src/main/activemq/wireformat/openwire/marshall/v4/ActiveMQDestinationMarshaller.h	
File Reference . . . . .	4294
7.189src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQDestinationMarshaller.h	
File Reference . . . . .	4295
7.190src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQDestinationMarshaller.h	
File Reference . . . . .	4296
7.191src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQMapMessageMarshaller.h	
File Reference . . . . .	4297
7.192src/main/activemq/wireformat/openwire/marshall/v2/ActiveMQMapMessageMarshaller.h	
File Reference . . . . .	4297
7.193src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQMapMessageMarshaller.h	
File Reference . . . . .	4298
7.194src/main/activemq/wireformat/openwire/marshall/v4/ActiveMQMapMessageMarshaller.h	
File Reference . . . . .	4299
7.195src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQMapMessageMarshaller.h	
File Reference . . . . .	4300
7.196src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQMapMessageMarshaller.h	
File Reference . . . . .	4300
7.197src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQMessageMarshaller.h	
File Reference . . . . .	4301

7.198src/main/activemq/wireformat/openwire/marshall/v2/ActiveMQMessageMarshaller.h	
File Reference . . . . .	4302
7.199src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQMessageMarshaller.h	
File Reference . . . . .	4303
7.200src/main/activemq/wireformat/openwire/marshall/v4/ActiveMQMessageMarshaller.h	
File Reference . . . . .	4303
7.201src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQMessageMarshaller.h	
File Reference . . . . .	4304
7.202src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQMessageMarshaller.h	
File Reference . . . . .	4305
7.203src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQObjectMessageMarshaller.h	
File Reference . . . . .	4306
7.204src/main/activemq/wireformat/openwire/marshall/v2/ActiveMQObjectMessageMarshaller.h	
File Reference . . . . .	4306
7.205src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQObjectMessageMarshaller.h	
File Reference . . . . .	4307
7.206src/main/activemq/wireformat/openwire/marshall/v4/ActiveMQObjectMessageMarshaller.h	
File Reference . . . . .	4308
7.207src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQObjectMessageMarshaller.h	
File Reference . . . . .	4309
7.208src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQObjectMessageMarshaller.h	
File Reference . . . . .	4309
7.209src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQQueueMarshaller.h	
File Reference . . . . .	4310
7.210src/main/activemq/wireformat/openwire/marshall/v2/ActiveMQQueueMarshaller.h	
File Reference . . . . .	4311
7.211src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQQueueMarshaller.h	
File Reference . . . . .	4312
7.212src/main/activemq/wireformat/openwire/marshall/v4/ActiveMQQueueMarshaller.h	
File Reference . . . . .	4312
7.213src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQQueueMarshaller.h	
File Reference . . . . .	4313
7.214src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQQueueMarshaller.h	
File Reference . . . . .	4314
7.215src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQStreamMessageMarshaller.h	
File Reference . . . . .	4315
7.216src/main/activemq/wireformat/openwire/marshall/v2/ActiveMQStreamMessageMarshaller.h	
File Reference . . . . .	4315
7.217src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQStreamMessageMarshaller.h	
File Reference . . . . .	4316
7.218src/main/activemq/wireformat/openwire/marshall/v4/ActiveMQStreamMessageMarshaller.h	
File Reference . . . . .	4317
7.219src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQStreamMessageMarshaller.h	
File Reference . . . . .	4318
7.220src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQStreamMessageMarshaller.h	
File Reference . . . . .	4319
7.221src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQTempDestinationMarshaller.h	
File Reference . . . . .	4319
7.222src/main/activemq/wireformat/openwire/marshall/v2/ActiveMQTempDestinationMarshaller.h	
File Reference . . . . .	4320

7.223src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQTempDestinationMarshaller.h	
File Reference . . . . .	4321
7.224src/main/activemq/wireformat/openwire/marshall/v4/ActiveMQTempDestinationMarshaller.h	
File Reference . . . . .	4322
7.225src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTempDestinationMarshaller.h	
File Reference . . . . .	4322
7.226src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQTempDestinationMarshaller.h	
File Reference . . . . .	4323
7.227src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQTempQueueMarshaller.h	
File Reference . . . . .	4324
7.228src/main/activemq/wireformat/openwire/marshall/v2/ActiveMQTempQueueMarshaller.h	
File Reference . . . . .	4325
7.229src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQTempQueueMarshaller.h	
File Reference . . . . .	4326
7.230src/main/activemq/wireformat/openwire/marshall/v4/ActiveMQTempQueueMarshaller.h	
File Reference . . . . .	4326
7.231src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTempQueueMarshaller.h	
File Reference . . . . .	4327
7.232src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQTempQueueMarshaller.h	
File Reference . . . . .	4328
7.233src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQTempTopicMarshaller.h	
File Reference . . . . .	4329
7.234src/main/activemq/wireformat/openwire/marshall/v2/ActiveMQTempTopicMarshaller.h	
File Reference . . . . .	4329
7.235src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQTempTopicMarshaller.h	
File Reference . . . . .	4330
7.236src/main/activemq/wireformat/openwire/marshall/v4/ActiveMQTempTopicMarshaller.h	
File Reference . . . . .	4331
7.237src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTempTopicMarshaller.h	
File Reference . . . . .	4332
7.238src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQTempTopicMarshaller.h	
File Reference . . . . .	4332
7.239src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQTextMessageMarshaller.h	
File Reference . . . . .	4333
7.240src/main/activemq/wireformat/openwire/marshall/v2/ActiveMQTextMessageMarshaller.h	
File Reference . . . . .	4334
7.241src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQTextMessageMarshaller.h	
File Reference . . . . .	4335
7.242src/main/activemq/wireformat/openwire/marshall/v4/ActiveMQTextMessageMarshaller.h	
File Reference . . . . .	4335
7.243src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTextMessageMarshaller.h	
File Reference . . . . .	4336
7.244src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQTextMessageMarshaller.h	
File Reference . . . . .	4337
7.245src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQTopicMarshaller.h	
File Reference . . . . .	4338
7.246src/main/activemq/wireformat/openwire/marshall/v2/ActiveMQTopicMarshaller.h	
File Reference . . . . .	4338
7.247src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQTopicMarshaller.h	
File Reference . . . . .	4339

7.248src/main/activemq/wireformat/openwire/marshall/v4/ActiveMQTopicMarshaller.h	
File Reference . . . . .	4340
7.249src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTopicMarshaller.h	
File Reference . . . . .	4341
7.250src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQTopicMarshaller.h	
File Reference . . . . .	4341
7.251src/main/activemq/wireformat/openwire/marshall/v1/BaseCommandMarshaller.h	
File Reference . . . . .	4342
7.252src/main/activemq/wireformat/openwire/marshall/v2/BaseCommandMarshaller.h	
File Reference . . . . .	4343
7.253src/main/activemq/wireformat/openwire/marshall/v3/BaseCommandMarshaller.h	
File Reference . . . . .	4344
7.254src/main/activemq/wireformat/openwire/marshall/v4/BaseCommandMarshaller.h	
File Reference . . . . .	4344
7.255src/main/activemq/wireformat/openwire/marshall/v5/BaseCommandMarshaller.h	
File Reference . . . . .	4345
7.256src/main/activemq/wireformat/openwire/marshall/v6/BaseCommandMarshaller.h	
File Reference . . . . .	4346
7.257src/main/activemq/wireformat/openwire/marshall/v1/BrokerIdMarshaller.h	
File Reference . . . . .	4347
7.258src/main/activemq/wireformat/openwire/marshall/v2/BrokerIdMarshaller.h	
File Reference . . . . .	4347
7.259src/main/activemq/wireformat/openwire/marshall/v3/BrokerIdMarshaller.h	
File Reference . . . . .	4348
7.260src/main/activemq/wireformat/openwire/marshall/v4/BrokerIdMarshaller.h	
File Reference . . . . .	4349
7.261src/main/activemq/wireformat/openwire/marshall/v5/BrokerIdMarshaller.h	
File Reference . . . . .	4349
7.262src/main/activemq/wireformat/openwire/marshall/v6/BrokerIdMarshaller.h	
File Reference . . . . .	4350
7.263src/main/activemq/wireformat/openwire/marshall/v1/BrokerInfoMarshaller.h	
File Reference . . . . .	4351
7.264src/main/activemq/wireformat/openwire/marshall/v2/BrokerInfoMarshaller.h	
File Reference . . . . .	4352
7.265src/main/activemq/wireformat/openwire/marshall/v3/BrokerInfoMarshaller.h	
File Reference . . . . .	4352
7.266src/main/activemq/wireformat/openwire/marshall/v4/BrokerInfoMarshaller.h	
File Reference . . . . .	4353
7.267src/main/activemq/wireformat/openwire/marshall/v5/BrokerInfoMarshaller.h	
File Reference . . . . .	4354
7.268src/main/activemq/wireformat/openwire/marshall/v6/BrokerInfoMarshaller.h	
File Reference . . . . .	4355
7.269src/main/activemq/wireformat/openwire/marshall/v1/ConnectionControlMarshaller.h	
File Reference . . . . .	4355
7.270src/main/activemq/wireformat/openwire/marshall/v2/ConnectionControlMarshaller.h	
File Reference . . . . .	4356
7.271src/main/activemq/wireformat/openwire/marshall/v3/ConnectionControlMarshaller.h	
File Reference . . . . .	4357
7.272src/main/activemq/wireformat/openwire/marshall/v4/ConnectionControlMarshaller.h	
File Reference . . . . .	4358



7.273src/main/activemq/wireformat/openwire/marshall/v5/ConnectionControlMarshaller.h	
File Reference . . . . .	4358
7.274src/main/activemq/wireformat/openwire/marshall/v6/ConnectionControlMarshaller.h	
File Reference . . . . .	4359
7.275src/main/activemq/wireformat/openwire/marshall/v1/ConnectionErrorMarshaller.h	
File Reference . . . . .	4360
7.276src/main/activemq/wireformat/openwire/marshall/v2/ConnectionErrorMarshaller.h	
File Reference . . . . .	4361
7.277src/main/activemq/wireformat/openwire/marshall/v3/ConnectionErrorMarshaller.h	
File Reference . . . . .	4361
7.278src/main/activemq/wireformat/openwire/marshall/v4/ConnectionErrorMarshaller.h	
File Reference . . . . .	4362
7.279src/main/activemq/wireformat/openwire/marshall/v5/ConnectionErrorMarshaller.h	
File Reference . . . . .	4363
7.280src/main/activemq/wireformat/openwire/marshall/v6/ConnectionErrorMarshaller.h	
File Reference . . . . .	4364
7.281src/main/activemq/wireformat/openwire/marshall/v1/ConnectionIdMarshaller.h	
File Reference . . . . .	4364
7.282src/main/activemq/wireformat/openwire/marshall/v2/ConnectionIdMarshaller.h	
File Reference . . . . .	4365
7.283src/main/activemq/wireformat/openwire/marshall/v3/ConnectionIdMarshaller.h	
File Reference . . . . .	4366
7.284src/main/activemq/wireformat/openwire/marshall/v4/ConnectionIdMarshaller.h	
File Reference . . . . .	4367
7.285src/main/activemq/wireformat/openwire/marshall/v5/ConnectionIdMarshaller.h	
File Reference . . . . .	4367
7.286src/main/activemq/wireformat/openwire/marshall/v6/ConnectionIdMarshaller.h	
File Reference . . . . .	4368
7.287src/main/activemq/wireformat/openwire/marshall/v1/ConnectionInfoMarshaller.h	
File Reference . . . . .	4369
7.288src/main/activemq/wireformat/openwire/marshall/v2/ConnectionInfoMarshaller.h	
File Reference . . . . .	4370
7.289src/main/activemq/wireformat/openwire/marshall/v3/ConnectionInfoMarshaller.h	
File Reference . . . . .	4370
7.290src/main/activemq/wireformat/openwire/marshall/v4/ConnectionInfoMarshaller.h	
File Reference . . . . .	4371
7.291src/main/activemq/wireformat/openwire/marshall/v5/ConnectionInfoMarshaller.h	
File Reference . . . . .	4372
7.292src/main/activemq/wireformat/openwire/marshall/v6/ConnectionInfoMarshaller.h	
File Reference . . . . .	4373
7.293src/main/activemq/wireformat/openwire/marshall/v1/ConsumerControlMarshaller.h	
File Reference . . . . .	4373
7.294src/main/activemq/wireformat/openwire/marshall/v2/ConsumerControlMarshaller.h	
File Reference . . . . .	4374
7.295src/main/activemq/wireformat/openwire/marshall/v3/ConsumerControlMarshaller.h	
File Reference . . . . .	4375
7.296src/main/activemq/wireformat/openwire/marshall/v4/ConsumerControlMarshaller.h	
File Reference . . . . .	4376
7.297src/main/activemq/wireformat/openwire/marshall/v5/ConsumerControlMarshaller.h	
File Reference . . . . .	4376

7.298src/main/activemq/wireformat/openwire/marshal/v6/ConsumerControlMarshaller.h	
File Reference . . . . .	4377
7.299src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h	
File Reference . . . . .	4378
7.300src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h	
File Reference . . . . .	4379
7.301src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h	
File Reference . . . . .	4379
7.302src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h	
File Reference . . . . .	4380
7.303src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h	
File Reference . . . . .	4381
7.304src/main/activemq/wireformat/openwire/marshal/v6/ConsumerIdMarshaller.h	
File Reference . . . . .	4382
7.305src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h	
File Reference . . . . .	4382
7.306src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h	
File Reference . . . . .	4383
7.307src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h	
File Reference . . . . .	4384
7.308src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h	
File Reference . . . . .	4385
7.309src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h	
File Reference . . . . .	4385
7.310src/main/activemq/wireformat/openwire/marshal/v6/ConsumerInfoMarshaller.h	
File Reference . . . . .	4386
7.311src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h	
File Reference . . . . .	4387
7.312src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h	
File Reference . . . . .	4388
7.313src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h	
File Reference . . . . .	4388
7.314src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h	
File Reference . . . . .	4389
7.315src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h	
File Reference . . . . .	4390
7.316src/main/activemq/wireformat/openwire/marshal/v6/ControlCommandMarshaller.h	
File Reference . . . . .	4391
7.317src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h	
File Reference . . . . .	4391
7.318src/main/activemq/wireformat/openwire/marshal/v2/DataArrayResponseMarshaller.h	
File Reference . . . . .	4392
7.319src/main/activemq/wireformat/openwire/marshal/v3/DataArrayResponseMarshaller.h	
File Reference . . . . .	4393
7.320src/main/activemq/wireformat/openwire/marshal/v4/DataArrayResponseMarshaller.h	
File Reference . . . . .	4394
7.321src/main/activemq/wireformat/openwire/marshal/v5/DataArrayResponseMarshaller.h	
File Reference . . . . .	4394
7.322src/main/activemq/wireformat/openwire/marshal/v6/DataArrayResponseMarshaller.h	
File Reference . . . . .	4395

7.323src/main/activemq/wireformat/openwire/marshall/v1/DataResponseMarshaller.h	
File Reference . . . . .	4396
7.324src/main/activemq/wireformat/openwire/marshall/v2/DataResponseMarshaller.h	
File Reference . . . . .	4397
7.325src/main/activemq/wireformat/openwire/marshall/v3/DataResponseMarshaller.h	
File Reference . . . . .	4397
7.326src/main/activemq/wireformat/openwire/marshall/v4/DataResponseMarshaller.h	
File Reference . . . . .	4398
7.327src/main/activemq/wireformat/openwire/marshall/v5/DataResponseMarshaller.h	
File Reference . . . . .	4399
7.328src/main/activemq/wireformat/openwire/marshall/v6/DataResponseMarshaller.h	
File Reference . . . . .	4400
7.329src/main/activemq/wireformat/openwire/marshall/v1/DestinationInfoMarshaller.h	
File Reference . . . . .	4400
7.330src/main/activemq/wireformat/openwire/marshall/v2/DestinationInfoMarshaller.h	
File Reference . . . . .	4401
7.331src/main/activemq/wireformat/openwire/marshall/v3/DestinationInfoMarshaller.h	
File Reference . . . . .	4402
7.332src/main/activemq/wireformat/openwire/marshall/v4/DestinationInfoMarshaller.h	
File Reference . . . . .	4403
7.333src/main/activemq/wireformat/openwire/marshall/v5/DestinationInfoMarshaller.h	
File Reference . . . . .	4403
7.334src/main/activemq/wireformat/openwire/marshall/v6/DestinationInfoMarshaller.h	
File Reference . . . . .	4404
7.335src/main/activemq/wireformat/openwire/marshall/v1/DiscoveryEventMarshaller.h	
File Reference . . . . .	4405
7.336src/main/activemq/wireformat/openwire/marshall/v2/DiscoveryEventMarshaller.h	
File Reference . . . . .	4406
7.337src/main/activemq/wireformat/openwire/marshall/v3/DiscoveryEventMarshaller.h	
File Reference . . . . .	4406
7.338src/main/activemq/wireformat/openwire/marshall/v4/DiscoveryEventMarshaller.h	
File Reference . . . . .	4407
7.339src/main/activemq/wireformat/openwire/marshall/v5/DiscoveryEventMarshaller.h	
File Reference . . . . .	4408
7.340src/main/activemq/wireformat/openwire/marshall/v6/DiscoveryEventMarshaller.h	
File Reference . . . . .	4409
7.341src/main/activemq/wireformat/openwire/marshall/v1/ExceptionResponseMarshaller.h	
File Reference . . . . .	4409
7.342src/main/activemq/wireformat/openwire/marshall/v2/ExceptionResponseMarshaller.h	
File Reference . . . . .	4410
7.343src/main/activemq/wireformat/openwire/marshall/v3/ExceptionResponseMarshaller.h	
File Reference . . . . .	4411
7.344src/main/activemq/wireformat/openwire/marshall/v4/ExceptionResponseMarshaller.h	
File Reference . . . . .	4412
7.345src/main/activemq/wireformat/openwire/marshall/v5/ExceptionResponseMarshaller.h	
File Reference . . . . .	4412
7.346src/main/activemq/wireformat/openwire/marshall/v6/ExceptionResponseMarshaller.h	
File Reference . . . . .	4413
7.347src/main/activemq/wireformat/openwire/marshall/v1/FlushCommandMarshaller.h	
File Reference . . . . .	4414

7.348src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h	
File Reference . . . . .	4415
7.349src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h	
File Reference . . . . .	4415
7.350src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h	
File Reference . . . . .	4416
7.351src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h	
File Reference . . . . .	4417
7.352src/main/activemq/wireformat/openwire/marshal/v6/FlushCommandMarshaller.h	
File Reference . . . . .	4418
7.353src/main/activemq/wireformat/openwire/marshal/v1/IntegerResponseMarshaller.h	
File Reference . . . . .	4418
7.354src/main/activemq/wireformat/openwire/marshal/v2/IntegerResponseMarshaller.h	
File Reference . . . . .	4419
7.355src/main/activemq/wireformat/openwire/marshal/v3/IntegerResponseMarshaller.h	
File Reference . . . . .	4420
7.356src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h	
File Reference . . . . .	4421
7.357src/main/activemq/wireformat/openwire/marshal/v5/IntegerResponseMarshaller.h	
File Reference . . . . .	4421
7.358src/main/activemq/wireformat/openwire/marshal/v6/IntegerResponseMarshaller.h	
File Reference . . . . .	4422
7.359src/main/activemq/wireformat/openwire/marshal/v1/JournalQueueAckMarshaller.h	
File Reference . . . . .	4423
7.360src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAckMarshaller.h	
File Reference . . . . .	4424
7.361src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h	
File Reference . . . . .	4424
7.362src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller.h	
File Reference . . . . .	4425
7.363src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h	
File Reference . . . . .	4426
7.364src/main/activemq/wireformat/openwire/marshal/v6/JournalQueueAckMarshaller.h	
File Reference . . . . .	4427
7.365src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h	
File Reference . . . . .	4427
7.366src/main/activemq/wireformat/openwire/marshal/v2/JournalTopicAckMarshaller.h	
File Reference . . . . .	4428
7.367src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h	
File Reference . . . . .	4429
7.368src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h	
File Reference . . . . .	4430
7.369src/main/activemq/wireformat/openwire/marshal/v5/JournalTopicAckMarshaller.h	
File Reference . . . . .	4430
7.370src/main/activemq/wireformat/openwire/marshal/v6/JournalTopicAckMarshaller.h	
File Reference . . . . .	4431
7.371src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshaller.h	
File Reference . . . . .	4432
7.372src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshaller.h	
File Reference . . . . .	4433

7.373src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h	
File Reference . . . . .	4433
7.374src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h	
File Reference . . . . .	4434
7.375src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMarshaller.h	
File Reference . . . . .	4435
7.376src/main/activemq/wireformat/openwire/marshal/v6/JournalTraceMarshaller.h	
File Reference . . . . .	4436
7.377src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h	
File Reference . . . . .	4436
7.378src/main/activemq/wireformat/openwire/marshal/v2/JournalTransactionMarshaller.h	
File Reference . . . . .	4437
7.379src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarshaller.h	
File Reference . . . . .	4438
7.380src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarshaller.h	
File Reference . . . . .	4439
7.381src/main/activemq/wireformat/openwire/marshal/v5/JournalTransactionMarshaller.h	
File Reference . . . . .	4439
7.382src/main/activemq/wireformat/openwire/marshal/v6/JournalTransactionMarshaller.h	
File Reference . . . . .	4440
7.383src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h	
File Reference . . . . .	4441
7.384src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h	
File Reference . . . . .	4442
7.385src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h	
File Reference . . . . .	4442
7.386src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h	
File Reference . . . . .	4443
7.387src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h	
File Reference . . . . .	4444
7.388src/main/activemq/wireformat/openwire/marshal/v6/KeepAliveInfoMarshaller.h	
File Reference . . . . .	4445
7.389src/main/activemq/wireformat/openwire/marshal/v1/LastPartialCommandMarshaller.h	
File Reference . . . . .	4445
7.390src/main/activemq/wireformat/openwire/marshal/v2/LastPartialCommandMarshaller.h	
File Reference . . . . .	4446
7.391src/main/activemq/wireformat/openwire/marshal/v3/LastPartialCommandMarshaller.h	
File Reference . . . . .	4447
7.392src/main/activemq/wireformat/openwire/marshal/v4/LastPartialCommandMarshaller.h	
File Reference . . . . .	4448
7.393src/main/activemq/wireformat/openwire/marshal/v5/LastPartialCommandMarshaller.h	
File Reference . . . . .	4448
7.394src/main/activemq/wireformat/openwire/marshal/v6/LastPartialCommandMarshaller.h	
File Reference . . . . .	4449
7.395src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h	
File Reference . . . . .	4450
7.396src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarshaller.h	
File Reference . . . . .	4451
7.397src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarshaller.h	
File Reference . . . . .	4451

7.398src/main/activemq/wireformat/openwire/marshall/v4/LocalTransactionIdMarshaller.h	
File Reference . . . . .	4452
7.399src/main/activemq/wireformat/openwire/marshall/v5/LocalTransactionIdMarshaller.h	
File Reference . . . . .	4453
7.400src/main/activemq/wireformat/openwire/marshall/v6/LocalTransactionIdMarshaller.h	
File Reference . . . . .	4454
7.401src/main/activemq/wireformat/openwire/marshall/v1/MarshallerFactory.h	
File Reference . . . . .	4454
7.402src/main/activemq/wireformat/openwire/marshall/v2/MarshallerFactory.h	
File Reference . . . . .	4455
7.403src/main/activemq/wireformat/openwire/marshall/v3/MarshallerFactory.h	
File Reference . . . . .	4455
7.404src/main/activemq/wireformat/openwire/marshall/v4/MarshallerFactory.h	
File Reference . . . . .	4456
7.405src/main/activemq/wireformat/openwire/marshall/v5/MarshallerFactory.h	
File Reference . . . . .	4456
7.406src/main/activemq/wireformat/openwire/marshall/v6/MarshallerFactory.h	
File Reference . . . . .	4457
7.407src/main/activemq/wireformat/openwire/marshall/v1/MessageAckMarshaller.h	
File Reference . . . . .	4458
7.408src/main/activemq/wireformat/openwire/marshall/v2/MessageAckMarshaller.h	
File Reference . . . . .	4458
7.409src/main/activemq/wireformat/openwire/marshall/v3/MessageAckMarshaller.h	
File Reference . . . . .	4459
7.410src/main/activemq/wireformat/openwire/marshall/v4/MessageAckMarshaller.h	
File Reference . . . . .	4460
7.411src/main/activemq/wireformat/openwire/marshall/v5/MessageAckMarshaller.h	
File Reference . . . . .	4461
7.412src/main/activemq/wireformat/openwire/marshall/v6/MessageAckMarshaller.h	
File Reference . . . . .	4461
7.413src/main/activemq/wireformat/openwire/marshall/v1/MessageDispatchMarshaller.h	
File Reference . . . . .	4462
7.414src/main/activemq/wireformat/openwire/marshall/v2/MessageDispatchMarshaller.h	
File Reference . . . . .	4463
7.415src/main/activemq/wireformat/openwire/marshall/v3/MessageDispatchMarshaller.h	
File Reference . . . . .	4464
7.416src/main/activemq/wireformat/openwire/marshall/v4/MessageDispatchMarshaller.h	
File Reference . . . . .	4464
7.417src/main/activemq/wireformat/openwire/marshall/v5/MessageDispatchMarshaller.h	
File Reference . . . . .	4465
7.418src/main/activemq/wireformat/openwire/marshall/v6/MessageDispatchMarshaller.h	
File Reference . . . . .	4466
7.419src/main/activemq/wireformat/openwire/marshall/v1/MessageDispatchNotificationMarshaller.h	
File Reference . . . . .	4467
7.420src/main/activemq/wireformat/openwire/marshall/v2/MessageDispatchNotificationMarshaller.h	
File Reference . . . . .	4467
7.421src/main/activemq/wireformat/openwire/marshall/v3/MessageDispatchNotificationMarshaller.h	
File Reference . . . . .	4468
7.422src/main/activemq/wireformat/openwire/marshall/v4/MessageDispatchNotificationMarshaller.h	
File Reference . . . . .	4469

7.423src/main/activemq/wireformat/openwire/marshall/v5/MessageDispatchNotificationMarshaller.h	
File Reference . . . . .	4470
7.424src/main/activemq/wireformat/openwire/marshall/v6/MessageDispatchNotificationMarshaller.h	
File Reference . . . . .	4470
7.425src/main/activemq/wireformat/openwire/marshall/v1/MessageIdMarshaller.h	
File Reference . . . . .	4471
7.426src/main/activemq/wireformat/openwire/marshall/v2/MessageIdMarshaller.h	
File Reference . . . . .	4472
7.427src/main/activemq/wireformat/openwire/marshall/v3/MessageIdMarshaller.h	
File Reference . . . . .	4473
7.428src/main/activemq/wireformat/openwire/marshall/v4/MessageIdMarshaller.h	
File Reference . . . . .	4473
7.429src/main/activemq/wireformat/openwire/marshall/v5/MessageIdMarshaller.h	
File Reference . . . . .	4474
7.430src/main/activemq/wireformat/openwire/marshall/v6/MessageIdMarshaller.h	
File Reference . . . . .	4475
7.431src/main/activemq/wireformat/openwire/marshall/v1/MessageMarshaller.h	
File Reference . . . . .	4476
7.432src/main/activemq/wireformat/openwire/marshall/v2/MessageMarshaller.h	
File Reference . . . . .	4476
7.433src/main/activemq/wireformat/openwire/marshall/v3/MessageMarshaller.h	
File Reference . . . . .	4477
7.434src/main/activemq/wireformat/openwire/marshall/v4/MessageMarshaller.h	
File Reference . . . . .	4478
7.435src/main/activemq/wireformat/openwire/marshall/v5/MessageMarshaller.h	
File Reference . . . . .	4479
7.436src/main/activemq/wireformat/openwire/marshall/v6/MessageMarshaller.h	
File Reference . . . . .	4479
7.437src/main/activemq/wireformat/openwire/marshall/v1/MessagePullMarshaller.h	
File Reference . . . . .	4480
7.438src/main/activemq/wireformat/openwire/marshall/v2/MessagePullMarshaller.h	
File Reference . . . . .	4481
7.439src/main/activemq/wireformat/openwire/marshall/v3/MessagePullMarshaller.h	
File Reference . . . . .	4482
7.440src/main/activemq/wireformat/openwire/marshall/v4/MessagePullMarshaller.h	
File Reference . . . . .	4482
7.441src/main/activemq/wireformat/openwire/marshall/v5/MessagePullMarshaller.h	
File Reference . . . . .	4483
7.442src/main/activemq/wireformat/openwire/marshall/v6/MessagePullMarshaller.h	
File Reference . . . . .	4484
7.443src/main/activemq/wireformat/openwire/marshall/v1/NetworkBridgeFilterMarshaller.h	
File Reference . . . . .	4485
7.444src/main/activemq/wireformat/openwire/marshall/v2/NetworkBridgeFilterMarshaller.h	
File Reference . . . . .	4485
7.445src/main/activemq/wireformat/openwire/marshall/v3/NetworkBridgeFilterMarshaller.h	
File Reference . . . . .	4486
7.446src/main/activemq/wireformat/openwire/marshall/v4/NetworkBridgeFilterMarshaller.h	
File Reference . . . . .	4487
7.447src/main/activemq/wireformat/openwire/marshall/v5/NetworkBridgeFilterMarshaller.h	
File Reference . . . . .	4488

7.448src/main/activemq/wireformat/openwire/marshall/v6/NetworkBridgeFilterMarshaller.h	
File Reference . . . . .	4488
7.449src/main/activemq/wireformat/openwire/marshall/v1/PartialCommandMarshaller.h	
File Reference . . . . .	4489
7.450src/main/activemq/wireformat/openwire/marshall/v2/PartialCommandMarshaller.h	
File Reference . . . . .	4490
7.451src/main/activemq/wireformat/openwire/marshall/v3/PartialCommandMarshaller.h	
File Reference . . . . .	4491
7.452src/main/activemq/wireformat/openwire/marshall/v4/PartialCommandMarshaller.h	
File Reference . . . . .	4491
7.453src/main/activemq/wireformat/openwire/marshall/v5/PartialCommandMarshaller.h	
File Reference . . . . .	4492
7.454src/main/activemq/wireformat/openwire/marshall/v6/PartialCommandMarshaller.h	
File Reference . . . . .	4493
7.455src/main/activemq/wireformat/openwire/marshall/v1/ProducerAckMarshaller.h	
File Reference . . . . .	4494
7.456src/main/activemq/wireformat/openwire/marshall/v2/ProducerAckMarshaller.h	
File Reference . . . . .	4494
7.457src/main/activemq/wireformat/openwire/marshall/v3/ProducerAckMarshaller.h	
File Reference . . . . .	4495
7.458src/main/activemq/wireformat/openwire/marshall/v4/ProducerAckMarshaller.h	
File Reference . . . . .	4496
7.459src/main/activemq/wireformat/openwire/marshall/v5/ProducerAckMarshaller.h	
File Reference . . . . .	4497
7.460src/main/activemq/wireformat/openwire/marshall/v6/ProducerAckMarshaller.h	
File Reference . . . . .	4497
7.461src/main/activemq/wireformat/openwire/marshall/v1/ProducerIdMarshaller.h	
File Reference . . . . .	4498
7.462src/main/activemq/wireformat/openwire/marshall/v2/ProducerIdMarshaller.h	
File Reference . . . . .	4499
7.463src/main/activemq/wireformat/openwire/marshall/v3/ProducerIdMarshaller.h	
File Reference . . . . .	4500
7.464src/main/activemq/wireformat/openwire/marshall/v4/ProducerIdMarshaller.h	
File Reference . . . . .	4500
7.465src/main/activemq/wireformat/openwire/marshall/v5/ProducerIdMarshaller.h	
File Reference . . . . .	4501
7.466src/main/activemq/wireformat/openwire/marshall/v6/ProducerIdMarshaller.h	
File Reference . . . . .	4502
7.467src/main/activemq/wireformat/openwire/marshall/v1/ProducerInfoMarshaller.h	
File Reference . . . . .	4503
7.468src/main/activemq/wireformat/openwire/marshall/v2/ProducerInfoMarshaller.h	
File Reference . . . . .	4503
7.469src/main/activemq/wireformat/openwire/marshall/v3/ProducerInfoMarshaller.h	
File Reference . . . . .	4504
7.470src/main/activemq/wireformat/openwire/marshall/v4/ProducerInfoMarshaller.h	
File Reference . . . . .	4505
7.471src/main/activemq/wireformat/openwire/marshall/v5/ProducerInfoMarshaller.h	
File Reference . . . . .	4506
7.472src/main/activemq/wireformat/openwire/marshall/v6/ProducerInfoMarshaller.h	
File Reference . . . . .	4506



7.473src/main/activemq/wireformat/openwire/marshall/v1/RemoveInfoMarshaller.h	
File Reference . . . . .	4507
7.474src/main/activemq/wireformat/openwire/marshall/v2/RemoveInfoMarshaller.h	
File Reference . . . . .	4508
7.475src/main/activemq/wireformat/openwire/marshall/v3/RemoveInfoMarshaller.h	
File Reference . . . . .	4509
7.476src/main/activemq/wireformat/openwire/marshall/v4/RemoveInfoMarshaller.h	
File Reference . . . . .	4509
7.477src/main/activemq/wireformat/openwire/marshall/v5/RemoveInfoMarshaller.h	
File Reference . . . . .	4510
7.478src/main/activemq/wireformat/openwire/marshall/v6/RemoveInfoMarshaller.h	
File Reference . . . . .	4511
7.479src/main/activemq/wireformat/openwire/marshall/v1/RemoveSubscriptionInfoMarshaller.h	
File Reference . . . . .	4512
7.480src/main/activemq/wireformat/openwire/marshall/v2/RemoveSubscriptionInfoMarshaller.h	
File Reference . . . . .	4512
7.481src/main/activemq/wireformat/openwire/marshall/v3/RemoveSubscriptionInfoMarshaller.h	
File Reference . . . . .	4513
7.482src/main/activemq/wireformat/openwire/marshall/v4/RemoveSubscriptionInfoMarshaller.h	
File Reference . . . . .	4514
7.483src/main/activemq/wireformat/openwire/marshall/v5/RemoveSubscriptionInfoMarshaller.h	
File Reference . . . . .	4515
7.484src/main/activemq/wireformat/openwire/marshall/v6/RemoveSubscriptionInfoMarshaller.h	
File Reference . . . . .	4515
7.485src/main/activemq/wireformat/openwire/marshall/v1/ReplayCommandMarshaller.h	
File Reference . . . . .	4516
7.486src/main/activemq/wireformat/openwire/marshall/v2/ReplayCommandMarshaller.h	
File Reference . . . . .	4517
7.487src/main/activemq/wireformat/openwire/marshall/v3/ReplayCommandMarshaller.h	
File Reference . . . . .	4518
7.488src/main/activemq/wireformat/openwire/marshall/v4/ReplayCommandMarshaller.h	
File Reference . . . . .	4518
7.489src/main/activemq/wireformat/openwire/marshall/v5/ReplayCommandMarshaller.h	
File Reference . . . . .	4519
7.490src/main/activemq/wireformat/openwire/marshall/v6/ReplayCommandMarshaller.h	
File Reference . . . . .	4520
7.491src/main/activemq/wireformat/openwire/marshall/v1/ResponseMarshaller.h	
File Reference . . . . .	4521
7.492src/main/activemq/wireformat/openwire/marshall/v2/ResponseMarshaller.h	
File Reference . . . . .	4521
7.493src/main/activemq/wireformat/openwire/marshall/v3/ResponseMarshaller.h	
File Reference . . . . .	4522
7.494src/main/activemq/wireformat/openwire/marshall/v4/ResponseMarshaller.h	
File Reference . . . . .	4523
7.495src/main/activemq/wireformat/openwire/marshall/v5/ResponseMarshaller.h	
File Reference . . . . .	4524
7.496src/main/activemq/wireformat/openwire/marshall/v6/ResponseMarshaller.h	
File Reference . . . . .	4524
7.497src/main/activemq/wireformat/openwire/marshall/v1/SessionIdMarshaller.h	
File Reference . . . . .	4525

7.498src/main/activemq/wireformat/openwire/marshall/v2/SessionIdMarshaller.h	
File Reference . . . . .	4526
7.499src/main/activemq/wireformat/openwire/marshall/v3/SessionIdMarshaller.h	
File Reference . . . . .	4526
7.500src/main/activemq/wireformat/openwire/marshall/v4/SessionIdMarshaller.h	
File Reference . . . . .	4527
7.501src/main/activemq/wireformat/openwire/marshall/v5/SessionIdMarshaller.h	
File Reference . . . . .	4528
7.502src/main/activemq/wireformat/openwire/marshall/v6/SessionIdMarshaller.h	
File Reference . . . . .	4529
7.503src/main/activemq/wireformat/openwire/marshall/v1/SessionInfoMarshaller.h	
File Reference . . . . .	4529
7.504src/main/activemq/wireformat/openwire/marshall/v2/SessionInfoMarshaller.h	
File Reference . . . . .	4530
7.505src/main/activemq/wireformat/openwire/marshall/v3/SessionInfoMarshaller.h	
File Reference . . . . .	4531
7.506src/main/activemq/wireformat/openwire/marshall/v4/SessionInfoMarshaller.h	
File Reference . . . . .	4532
7.507src/main/activemq/wireformat/openwire/marshall/v5/SessionInfoMarshaller.h	
File Reference . . . . .	4532
7.508src/main/activemq/wireformat/openwire/marshall/v6/SessionInfoMarshaller.h	
File Reference . . . . .	4533
7.509src/main/activemq/wireformat/openwire/marshall/v1/ShutdownInfoMarshaller.h	
File Reference . . . . .	4534
7.510src/main/activemq/wireformat/openwire/marshall/v2/ShutdownInfoMarshaller.h	
File Reference . . . . .	4535
7.511src/main/activemq/wireformat/openwire/marshall/v3/ShutdownInfoMarshaller.h	
File Reference . . . . .	4535
7.512src/main/activemq/wireformat/openwire/marshall/v4/ShutdownInfoMarshaller.h	
File Reference . . . . .	4536
7.513src/main/activemq/wireformat/openwire/marshall/v5/ShutdownInfoMarshaller.h	
File Reference . . . . .	4537
7.514src/main/activemq/wireformat/openwire/marshall/v6/ShutdownInfoMarshaller.h	
File Reference . . . . .	4538
7.515src/main/activemq/wireformat/openwire/marshall/v1/SubscriptionInfoMarshaller.h	
File Reference . . . . .	4538
7.516src/main/activemq/wireformat/openwire/marshall/v2/SubscriptionInfoMarshaller.h	
File Reference . . . . .	4539
7.517src/main/activemq/wireformat/openwire/marshall/v3/SubscriptionInfoMarshaller.h	
File Reference . . . . .	4540
7.518src/main/activemq/wireformat/openwire/marshall/v4/SubscriptionInfoMarshaller.h	
File Reference . . . . .	4541
7.519src/main/activemq/wireformat/openwire/marshall/v5/SubscriptionInfoMarshaller.h	
File Reference . . . . .	4541
7.520src/main/activemq/wireformat/openwire/marshall/v6/SubscriptionInfoMarshaller.h	
File Reference . . . . .	4542
7.521src/main/activemq/wireformat/openwire/marshall/v1/TransactionIdMarshaller.h	
File Reference . . . . .	4543
7.522src/main/activemq/wireformat/openwire/marshall/v2/TransactionIdMarshaller.h	
File Reference . . . . .	4544

7.523src/main/activemq/wireformat/openwire/marshall/v3/TransactionIdMarshaller.h	
File Reference . . . . .	4544
7.524src/main/activemq/wireformat/openwire/marshall/v4/TransactionIdMarshaller.h	
File Reference . . . . .	4545
7.525src/main/activemq/wireformat/openwire/marshall/v5/TransactionIdMarshaller.h	
File Reference . . . . .	4546
7.526src/main/activemq/wireformat/openwire/marshall/v6/TransactionIdMarshaller.h	
File Reference . . . . .	4547
7.527src/main/activemq/wireformat/openwire/marshall/v1/TransactionInfoMarshaller.h	
File Reference . . . . .	4547
7.528src/main/activemq/wireformat/openwire/marshall/v2/TransactionInfoMarshaller.h	
File Reference . . . . .	4548
7.529src/main/activemq/wireformat/openwire/marshall/v3/TransactionInfoMarshaller.h	
File Reference . . . . .	4549
7.530src/main/activemq/wireformat/openwire/marshall/v4/TransactionInfoMarshaller.h	
File Reference . . . . .	4550
7.531src/main/activemq/wireformat/openwire/marshall/v5/TransactionInfoMarshaller.h	
File Reference . . . . .	4550
7.532src/main/activemq/wireformat/openwire/marshall/v6/TransactionInfoMarshaller.h	
File Reference . . . . .	4551
7.533src/main/activemq/wireformat/openwire/marshall/v1/WireFormatInfoMarshaller.h	
File Reference . . . . .	4552
7.534src/main/activemq/wireformat/openwire/marshall/v2/WireFormatInfoMarshaller.h	
File Reference . . . . .	4553
7.535src/main/activemq/wireformat/openwire/marshall/v3/WireFormatInfoMarshaller.h	
File Reference . . . . .	4553
7.536src/main/activemq/wireformat/openwire/marshall/v4/WireFormatInfoMarshaller.h	
File Reference . . . . .	4554
7.537src/main/activemq/wireformat/openwire/marshall/v5/WireFormatInfoMarshaller.h	
File Reference . . . . .	4555
7.538src/main/activemq/wireformat/openwire/marshall/v6/WireFormatInfoMarshaller.h	
File Reference . . . . .	4556
7.539src/main/activemq/wireformat/openwire/marshall/v1/XATransactionIdMarshaller.h	
File Reference . . . . .	4556
7.540src/main/activemq/wireformat/openwire/marshall/v2/XATransactionIdMarshaller.h	
File Reference . . . . .	4557
7.541src/main/activemq/wireformat/openwire/marshall/v3/XATransactionIdMarshaller.h	
File Reference . . . . .	4558
7.542src/main/activemq/wireformat/openwire/marshall/v4/XATransactionIdMarshaller.h	
File Reference . . . . .	4559
7.543src/main/activemq/wireformat/openwire/marshall/v5/XATransactionIdMarshaller.h	
File Reference . . . . .	4559
7.544src/main/activemq/wireformat/openwire/marshall/v6/XATransactionIdMarshaller.h	
File Reference . . . . .	4560
7.545src/main/activemq/wireformat/openwire/OpenWireFormat.h File Ref-	
erence . . . . .	4561
7.546src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h File	
Reference . . . . .	4562
7.547src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h	
File Reference . . . . .	4562

7.548src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h File Reference . . . . .	4563
7.549src/main/activemq/wireformat/openwire/utils/BooleanStream.h File Reference . . . . .	4564
7.550src/main/activemq/wireformat/openwire/utils/HexTable.h File Reference . . . . .	4564
7.551src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h File Reference . . . . .	4565
7.552src/main/activemq/wireformat/stomp/StompCommandConstants.h File Reference . . . . .	4565
7.553src/main/activemq/wireformat/stomp/StompFrame.h File Reference . . . . .	4566
7.554src/main/activemq/wireformat/stomp/StompHelper.h File Reference . . . . .	4567
7.555src/main/activemq/wireformat/stomp/StompWireFormat.h File Reference . . . . .	4567
7.556src/main/activemq/wireformat/stomp/StompWireFormatFactory.h File Reference . . . . .	4568
7.557src/main/activemq/wireformat/WireFormat.h File Reference . . . . .	4569
7.558src/main/activemq/wireformat/WireFormatFactory.h File Reference . . . . .	4569
7.559src/main/activemq/wireformat/WireFormatNegotiator.h File Reference . . . . .	4570
7.560src/main/activemq/wireformat/WireFormatRegistry.h File Reference . . . . .	4570
7.561src/main/cms/BytesMessage.h File Reference . . . . .	4571
7.562src/main/cms/Closeable.h File Reference . . . . .	4572
7.563src/main/decaf/io/Closeable.h File Reference . . . . .	4572
7.564src/main/cms/CMSException.h File Reference . . . . .	4573
7.565src/main/cms/CMSProperties.h File Reference . . . . .	4573
7.566src/main/cms/CMSSecurityException.h File Reference . . . . .	4574
7.567src/main/cms/Connection.h File Reference . . . . .	4574
7.568src/main/cms/ConnectionFactory.h File Reference . . . . .	4575
7.569src/main/cms/ConnectionMetaData.h File Reference . . . . .	4575
7.570src/main/cms/DeliveryMode.h File Reference . . . . .	4576
7.571src/main/cms/Destination.h File Reference . . . . .	4576
7.572src/main/cms/ExceptionListener.h File Reference . . . . .	4576
7.573src/main/cms/IllegalStateException.h File Reference . . . . .	4577
7.574src/main/decaf/lang/exceptions/IllegalStateException.h File Reference . . . . .	4577
7.575src/main/cms/InvalidClientIdException.h File Reference . . . . .	4578
7.576src/main/cms/InvalidDestinationException.h File Reference . . . . .	4578
7.577src/main/cms/InvalidSelectorException.h File Reference . . . . .	4579
7.578src/main/cms/MapMessage.h File Reference . . . . .	4579
7.579src/main/cms/MessageConsumer.h File Reference . . . . .	4580
7.580src/main/cms/MessageEnumeration.h File Reference . . . . .	4580
7.581src/main/cms/MessageEOFException.h File Reference . . . . .	4581
7.582src/main/cms/MessageFormatException.h File Reference . . . . .	4581
7.583src/main/cms/MessageListener.h File Reference . . . . .	4582
7.584src/main/cms/MessageNotReadableException.h File Reference . . . . .	4582
7.585src/main/cms/MessageNotWritableException.h File Reference . . . . .	4583
7.586src/main/cms/MessageProducer.h File Reference . . . . .	4583
7.587src/main/cms/ObjectMessage.h File Reference . . . . .	4584
7.588src/main/cms/Queue.h File Reference . . . . .	4584
7.589src/main/decaf/util/Queue.h File Reference . . . . .	4585
7.590src/main/cms/QueueBrowser.h File Reference . . . . .	4585
7.591src/main/cms/Session.h File Reference . . . . .	4586

7.592src/main/cms/Startable.h File Reference . . . . .	4587
7.593src/main/cms/Stopable.h File Reference . . . . .	4587
7.594src/main/cms/StreamMessage.h File Reference . . . . .	4587
7.595src/main/cms/TemporaryQueue.h File Reference . . . . .	4588
7.596src/main/cms/TemporaryTopic.h File Reference . . . . .	4588
7.597src/main/cms/TextMessage.h File Reference . . . . .	4589
7.598src/main/cms/Topic.h File Reference . . . . .	4589
7.599src/main/cms/UnsupportedOperationException.h File Reference . . .	4590
7.600src/main/decaf/lang/exceptions/UnsupportedOperationException.h File Reference . . . . .	4590
7.601src/main/decaf/internal/AprPool.h File Reference . . . . .	4591
7.602src/main/decaf/internal/DecafRuntime.h File Reference . . . . .	4591
7.603src/main/decaf/internal/io/StandardErrorOutputStream.h File Reference	4592
7.604src/main/decaf/internal/io/StandardInputStream.h File Reference . . .	4592
7.605src/main/decaf/internal/io/StandardOutputStream.h File Reference . .	4593
7.606src/main/decaf/internal/net/DefaultServerSocketFactory.h File Reference	4593
7.607src/main/decaf/internal/net/DefaultSocketFactory.h File Reference . .	4594
7.608src/main/decaf/internal/net/Network.h File Reference . . . . .	4594
7.609src/main/decaf/internal/net/SocketFileDescriptor.h File Reference . .	4595
7.610src/main/decaf/internal/net/ssl/DefaultSSLContext.h File Reference . .	4595
7.611src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h File Ref- erence . . . . .	4596
7.612src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h File Refer- ence . . . . .	4597
7.613src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h File Ref- erence . . . . .	4597
7.614src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h File Ref- erence . . . . .	4598
7.615src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h File Ref- erence . . . . .	4598
7.616src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h File Reference . . . . .	4599
7.617src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h File Reference	4600
7.618src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h File Reference . . . . .	4600
7.619src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h File Ref- erence . . . . .	4601
7.620src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h File Reference . . . . .	4601
7.621src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h File Reference . . . . .	4602
7.622src/main/decaf/internal/net/tcp/TcpSocket.h File Reference . . . . .	4603
7.623src/main/decaf/internal/net/tcp/TcpSocketInputStream.h File Reference	4603
7.624src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h File Reference	4604
7.625src/main/decaf/internal/net/URIEncoderDecoder.h File Reference . .	4604
7.626src/main/decaf/internal/net/URIHelper.h File Reference . . . . .	4605
7.627src/main/decaf/internal/net/URIType.h File Reference . . . . .	4605
7.628src/main/decaf/internal/nio/BufferFactory.h File Reference . . . . .	4606
7.629src/main/decaf/internal/nio/ByteBuffer.h File Reference . . . . .	4607
7.630src/main/decaf/internal/nio/CharArrayBuffer.h File Reference . . . .	4607

7.631src/main/decaf/internal/nio/DoubleArrayBuffer.h File Reference . . .	4608
7.632src/main/decaf/internal/nio/FloatArrayBuffer.h File Reference . . . .	4609
7.633src/main/decaf/internal/nio/IntArrayBuffer.h File Reference . . . . .	4609
7.634src/main/decaf/internal/nio/LongArrayBuffer.h File Reference . . . .	4610
7.635src/main/decaf/internal/nio/ShortArrayBuffer.h File Reference . . . .	4610
7.636src/main/decaf/internal/security/unix/SecureRandomImpl.h File Refer- ence . . . . .	4611
7.637src/main/decaf/internal/security/windows/SecureRandomImpl.h File Ref- erence . . . . .	4612
7.638src/main/decaf/internal/util/ByteArrayAdapter.h File Reference . . . .	4612
7.639src/main/decaf/internal/util/concurrent/ConditionImpl.h File Reference	4613
7.640src/main/decaf/internal/util/concurrent/MutexImpl.h File Reference .	4613
7.641src/main/decaf/internal/util/concurrent/SynchronizableImpl.h File Ref- erence . . . . .	4614
7.642src/main/decaf/internal/util/concurrent/Transferer.h File Reference . .	4614
7.643src/main/decaf/internal/util/concurrent/TransferQueue.h File Reference	4615
7.644src/main/decaf/internal/util/concurrent/TransferStack.h File Reference	4616
7.645src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h File Ref- erence . . . . .	4616
7.646src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h File Reference . . . . .	4617
7.647src/main/decaf/internal/util/concurrent/unix/MutexHandle.h File Ref- erence . . . . .	4617
7.648src/main/decaf/internal/util/concurrent/windows/MutexHandle.h File Ref- erence . . . . .	4617
7.649src/main/decaf/internal/util/GenericResource.h File Reference . . . .	4618
7.650src/main/decaf/internal/util/HexStringParser.h File Reference . . . .	4618
7.651src/main/decaf/internal/util/Resource.h File Reference . . . . .	4619
7.652src/main/decaf/internal/util/TimerTaskHeap.h File Reference . . . .	4619
7.653src/main/decaf/internal/util/zip/crc32.h File Reference . . . . .	4620
7.653.1 Variable Documentation . . . . .	4620
7.653.1.1 crc_table . . . . .	4620
7.654src/main/decaf/internal/util/zip/deflate.h File Reference . . . . .	4620
7.654.1 Define Documentation . . . . .	4621
7.654.1.1 _tr_tally_dist . . . . .	4621
7.654.1.2 _tr_tally_lit . . . . .	4622
7.654.1.3 BL_CODES . . . . .	4623
7.654.1.4 BUSY_STATE . . . . .	4623
7.654.1.5 Code . . . . .	4623
7.654.1.6 COMMENT_STATE . . . . .	4623
7.654.1.7 d_code . . . . .	4623
7.654.1.8 D_CODES . . . . .	4623
7.654.1.9 Dad . . . . .	4623
7.654.1.10EXTRA_STATE . . . . .	4623
7.654.1.11IFINISH_STATE . . . . .	4623
7.654.1.12Freq . . . . .	4623
7.654.1.13GZIP . . . . .	4623
7.654.1.14HCRC_STATE . . . . .	4623
7.654.1.15HEAP_SIZE . . . . .	4623
7.654.1.16INIT_STATE . . . . .	4623

7.654.1.17	L_CODES	4623
7.654.1.18	Len	4623
7.654.1.19	LENGTH_CODES	4623
7.654.1.20	LITERALS	4623
7.654.1.21	IMAX_BITS	4623
7.654.1.22	MAX_DIST	4623
7.654.1.23	max_insert_length	4623
7.654.1.24	MIN_LOOKAHEAD	4623
7.654.1.25	NAME_STATE	4623
7.654.1.26	put_byte	4623
7.654.1.27	WIN_INIT	4623
7.654.2	Typedef Documentation	4623
7.654.2.1	ct_data	4623
7.654.2.2	deflate_state	4623
7.654.2.3	IPos	4623
7.654.2.4	Pos	4623
7.654.2.5	Posf	4623
7.654.2.6	static_tree_desc	4623
7.654.2.7	tree_desc	4623
7.654.3	Function Documentation	4623
7.654.3.1	OF	4623
7.654.3.2	OF	4623
7.654.3.3	OF	4623
7.654.4	Variable Documentation	4623
7.654.4.1	_dist_code	4623
7.654.4.2	_length_code	4623
7.655	src/main/decaf/internal/util/zip/gzguts.h File Reference	4623
7.655.1	Define Documentation	4625
7.655.1.1	COPY	4625
7.655.1.2	GT_OFF	4625
7.655.1.3	GZ_APPEND	4625
7.655.1.4	GZ_NONE	4625
7.655.1.5	GZ_READ	4625
7.655.1.6	GZ_WRITE	4625
7.655.1.7	GZBUFSIZE	4625
7.655.1.8	GZIP	4625
7.655.1.9	local	4625
7.655.1.10	LOOK	4625
7.655.1.11	IZLIB_INTERNAL	4625
7.655.1.12	zstrerror	4625
7.655.2	Typedef Documentation	4625
7.655.2.1	gz_statep	4625
7.655.3	Function Documentation	4625
7.655.3.1	OF	4625
7.655.3.2	OF	4625
7.655.3.3	OF	4625
7.655.3.4	OF	4625
7.655.3.5	OF	4625
7.655.3.6	OF	4625
7.655.3.7	OF	4625

7.656src/main/decaf/internal/util/zip/inffast.h File Reference . . . . .	4625
7.656.1 Function Documentation . . . . .	4626
7.656.1.1 OF . . . . .	4626
7.657src/main/decaf/internal/util/zip/inffixed.h File Reference . . . . .	4626
7.658src/main/decaf/internal/util/zip/inflate.h File Reference . . . . .	4626
7.658.1 Define Documentation . . . . .	4626
7.658.1.1 GUNZIP . . . . .	4626
7.658.2 Enumeration Type Documentation . . . . .	4626
7.658.2.1 inflate_mode . . . . .	4626
7.659src/main/decaf/internal/util/zip/inftrees.h File Reference . . . . .	4627
7.659.1 Define Documentation . . . . .	4628
7.659.1.1 ENOUGH . . . . .	4628
7.659.1.2 ENOUGH_DIST . . . . .	4628
7.659.1.3 ENOUGH_LENS . . . . .	4628
7.659.2 Enumeration Type Documentation . . . . .	4628
7.659.2.1 codetype . . . . .	4628
7.659.3 Function Documentation . . . . .	4628
7.659.3.1 OF . . . . .	4628
7.660src/main/decaf/internal/util/zip/trees.h File Reference . . . . .	4628
7.660.1 Variable Documentation . . . . .	4629
7.660.1.1 _dist_code . . . . .	4629
7.660.1.2 _length_code . . . . .	4629
7.660.1.3 base_dist . . . . .	4630
7.660.1.4 base_length . . . . .	4630
7.660.1.5 static_dtree . . . . .	4630
7.660.1.6 static_ltree . . . . .	4630
7.661src/main/decaf/internal/util/zip/zconf.h File Reference . . . . .	4630
7.661.1 Define Documentation . . . . .	4632
7.661.1.1 const . . . . .	4632
7.661.1.2 FAR . . . . .	4632
7.661.1.3 MAX_MEM_LEVEL . . . . .	4632
7.661.1.4 MAX_WBITS . . . . .	4632
7.661.1.5 OF . . . . .	4632
7.661.1.6 SEEK_CUR . . . . .	4632
7.661.1.7 SEEK_END . . . . .	4632
7.661.1.8 SEEK_SET . . . . .	4632
7.661.1.9 z_off64_t . . . . .	4632
7.661.1.10z_off_t . . . . .	4632
7.661.1.11ZEXPORT . . . . .	4632
7.661.1.12ZEXPORTVA . . . . .	4632
7.661.1.13ZEXTERN . . . . .	4632
7.661.2 Typedef Documentation . . . . .	4632
7.661.2.1 Byte . . . . .	4632
7.661.2.2 Bytef . . . . .	4632
7.661.2.3 charf . . . . .	4632
7.661.2.4 intf . . . . .	4632
7.661.2.5 uInt . . . . .	4632
7.661.2.6 uIntf . . . . .	4632
7.661.2.7 uLong . . . . .	4632
7.661.2.8 uLongf . . . . .	4632



7.661.2.9 voidp . . . . .	4632
7.661.2.10voidpc . . . . .	4632
7.661.2.1 lvoidpf . . . . .	4632
7.662src/main/decaf/internal/util/zip/zlib.h File Reference . . . . .	4632
7.662.1 Define Documentation . . . . .	4635
7.662.1.1 deflateInit . . . . .	4635
7.662.1.2 deflateInit2 . . . . .	4635
7.662.1.3 inflateBackInit . . . . .	4635
7.662.1.4 inflateInit . . . . .	4636
7.662.1.5 inflateInit2 . . . . .	4636
7.662.1.6 Z_ASCII . . . . .	4636
7.662.1.7 Z_BEST_COMPRESSION . . . . .	4636
7.662.1.8 Z_BEST_SPEED . . . . .	4636
7.662.1.9 Z_BINARY . . . . .	4636
7.662.1.10Z_BLOCK . . . . .	4636
7.662.1.11Z_BUF_ERROR . . . . .	4636
7.662.1.12Z_DATA_ERROR . . . . .	4636
7.662.1.13Z_DEFAULT_COMPRESSION . . . . .	4636
7.662.1.14Z_DEFAULT_STRATEGY . . . . .	4636
7.662.1.15Z_DEFLATED . . . . .	4636
7.662.1.16Z_ERRNO . . . . .	4636
7.662.1.17Z_FILTERED . . . . .	4636
7.662.1.18Z_FINISH . . . . .	4636
7.662.1.19Z_FIXED . . . . .	4636
7.662.1.20Z_FULL_FLUSH . . . . .	4636
7.662.1.21Z_HUFFMAN_ONLY . . . . .	4636
7.662.1.22Z_MEM_ERROR . . . . .	4636
7.662.1.23Z_NEED_DICT . . . . .	4636
7.662.1.24Z_NO_COMPRESSION . . . . .	4636
7.662.1.25Z_NO_FLUSH . . . . .	4636
7.662.1.26Z_NULL . . . . .	4636
7.662.1.27Z_OK . . . . .	4636
7.662.1.28Z_PARTIAL_FLUSH . . . . .	4636
7.662.1.29Z_RLE . . . . .	4636
7.662.1.30Z_STREAM_END . . . . .	4636
7.662.1.31Z_STREAM_ERROR . . . . .	4636
7.662.1.32Z_SYNC_FLUSH . . . . .	4636
7.662.1.33Z_TEXT . . . . .	4636
7.662.1.34Z_TREES . . . . .	4636
7.662.1.35Z_UNKNOWN . . . . .	4636
7.662.1.36Z_VERSION_ERROR . . . . .	4636
7.662.1.37ZLIB_VER_MAJOR . . . . .	4636
7.662.1.38ZLIB_VER_MINOR . . . . .	4636
7.662.1.39ZLIB_VER_REVISION . . . . .	4636
7.662.1.40ZLIB_VER_SUBREVISION . . . . .	4636
7.662.1.41ZLIB_VERNUM . . . . .	4636
7.662.1.42zlib_version . . . . .	4636
7.662.1.43ZLIB_VERSION . . . . .	4636
7.662.2 Typedef Documentation . . . . .	4636
7.662.2.1 gz_header . . . . .	4636

7.662.2.2 gz_headerp . . . . .	4636
7.662.2.3 gzFile . . . . .	4636
7.662.2.4 OF . . . . .	4636
7.662.2.5 z_stream . . . . .	4636
7.662.2.6 z_streamp . . . . .	4636
7.662.3 Function Documentation . . . . .	4636
7.662.3.1 OF . . . . .	4636
7.662.3.2 OF . . . . .	4636
7.662.3.3 OF . . . . .	4636
7.662.3.4 OF . . . . .	4636
7.662.3.5 OF . . . . .	4636
7.662.3.6 OF . . . . .	4636
7.662.3.7 OF . . . . .	4636
7.662.3.8 OF . . . . .	4636
7.662.3.9 OF . . . . .	4636
7.662.3.10OF . . . . .	4636
7.662.3.11OF . . . . .	4636
7.662.3.12OF . . . . .	4636
7.662.3.13OF . . . . .	4636
7.662.3.14OF . . . . .	4636
7.662.3.15OF . . . . .	4636
7.662.3.16OF . . . . .	4636
7.662.3.17OF . . . . .	4636
7.662.3.18OF . . . . .	4636
7.662.3.19OF . . . . .	4636
7.662.3.20OF . . . . .	4636
7.662.3.21OF . . . . .	4636
7.662.3.22OF . . . . .	4636
7.662.3.23OF . . . . .	4636
7.662.3.24OF . . . . .	4636
7.662.3.25OF . . . . .	4636
7.662.3.26OF . . . . .	4636
7.662.3.27OF . . . . .	4636
7.662.3.28OF . . . . .	4636
7.662.3.29OF . . . . .	4636
7.662.3.30OF . . . . .	4636
7.662.3.31OF . . . . .	4636
7.662.3.32OF . . . . .	4636
7.662.3.33OF . . . . .	4636
7.662.3.34OF . . . . .	4636
7.662.3.35OF . . . . .	4636
7.662.3.36OF . . . . .	4636
7.662.3.37OF . . . . .	4636
7.662.3.38OF . . . . .	4636
7.662.3.39OF . . . . .	4636
7.662.3.40OF . . . . .	4636
7.662.3.41OF . . . . .	4636
7.662.3.42OF . . . . .	4636
7.663src/main/decaf/internal/util/zip/zutil.h File Reference . . . . .	4636
7.663.1 Define Documentation . . . . .	4639

7.663.1.1	Assert	4639
7.663.1.2	DEF_MEM_LEVEL	4639
7.663.1.3	DEF_WBITS	4639
7.663.1.4	DYN_TREES	4639
7.663.1.5	ERR_MSG	4639
7.663.1.6	ERR_RETURN	4639
7.663.1.7	F_OPEN	4639
7.663.1.8	local	4639
7.663.1.9	MAX_MATCH	4639
7.663.1.10	MIN_MATCH	4639
7.663.1.11	IOS_CODE	4639
7.663.1.12	PRESET_DICT	4639
7.663.1.13	STATIC_TREES	4639
7.663.1.14	STORED_BLOCK	4639
7.663.1.15	Trace	4639
7.663.1.16	Tracec	4639
7.663.1.17	Tracecv	4639
7.663.1.18	Tracev	4639
7.663.1.19	Tracevv	4639
7.663.1.20	TRY_FREE	4639
7.663.1.21	ZALLOC	4639
7.663.1.22	ZFREE	4639
7.663.1.23	ZLIB_INTERNAL	4639
7.663.2	Typedef Documentation	4639
7.663.2.1	uch	4639
7.663.2.2	uchf	4639
7.663.2.3	ulg	4639
7.663.2.4	ush	4639
7.663.2.5	ushf	4639
7.663.3	Function Documentation	4639
7.663.3.1	OF	4639
7.663.3.2	OF	4639
7.663.3.3	OF	4639
7.663.3.4	OF	4639
7.663.3.5	OF	4639
7.663.3.6	OF	4639
7.663.4	Variable Documentation	4639
7.663.4.1	z_errmsg	4639
7.664	src/main/decaf/io/BlockingByteArrayInputStream.h File Reference	4639
7.665	src/main/decaf/io/BufferedInputStream.h File Reference	4640
7.666	src/main/decaf/io/BufferedOutputStream.h File Reference	4640
7.667	src/main/decaf/io/ByteArrayInputStream.h File Reference	4641
7.668	src/main/decaf/io/ByteArrayOutputStream.h File Reference	4641
7.669	src/main/decaf/io/DataInput.h File Reference	4642
7.670	src/main/decaf/io/DataInputStream.h File Reference	4642
7.671	src/main/decaf/io/DataOutput.h File Reference	4643
7.672	src/main/decaf/io/DataOutputStream.h File Reference	4644
7.673	src/main/decaf/io/EOFException.h File Reference	4644
7.674	src/main/decaf/io/FileDescriptor.h File Reference	4645
7.675	src/main/decaf/io/FilterInputStream.h File Reference	4645

7.676src/main/decaf/io/FilterOutputStream.h File Reference . . . . .	4646
7.677src/main/decaf/io/Flushable.h File Reference . . . . .	4646
7.678src/main/decaf/io/InputStream.h File Reference . . . . .	4647
7.679src/main/decaf/io/InputStreamReader.h File Reference . . . . .	4647
7.680src/main/decaf/io/InterruptedIOException.h File Reference . . . . .	4648
7.681src/main/decaf/io/IOException.h File Reference . . . . .	4648
7.682src/main/decaf/io/OutputStream.h File Reference . . . . .	4649
7.683src/main/decaf/io/OutputStreamWriter.h File Reference . . . . .	4649
7.684src/main/decaf/io/PushbackInputStream.h File Reference . . . . .	4650
7.685src/main/decaf/io/Reader.h File Reference . . . . .	4650
7.686src/main/decaf/io/UnsupportedEncodingException.h File Reference . . . . .	4651
7.687src/main/decaf/io/UTFDataFormatException.h File Reference . . . . .	4651
7.688src/main/decaf/io/Writer.h File Reference . . . . .	4652
7.689src/main/decaf/lang/Appendable.h File Reference . . . . .	4652
7.690src/main/decaf/lang/ArrayPointer.h File Reference . . . . .	4653
7.691src/main/decaf/lang/Boolean.h File Reference . . . . .	4654
7.692src/main/decaf/lang/Byte.h File Reference . . . . .	4654
7.693src/main/decaf/lang/Character.h File Reference . . . . .	4655
7.694src/main/decaf/lang/CharSequence.h File Reference . . . . .	4655
7.695src/main/decaf/lang/Comparable.h File Reference . . . . .	4656
7.696src/main/decaf/lang/Double.h File Reference . . . . .	4656
7.697src/main/decaf/lang/Exception.h File Reference . . . . .	4657
7.698src/main/decaf/lang/exceptions/ClassCastException.h File Reference . . . . .	4657
7.699src/main/decaf/lang/exceptions/IllegalArgumentException.h File Reference . . . . .	4658
7.700src/main/decaf/lang/exceptions/IllegalMonitorStateException.h File Reference . . . . .	4658
7.701src/main/decaf/lang/exceptions/IllegalThreadStateException.h File Reference . . . . .	4659
7.702src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h File Reference . . . . .	4659
7.703src/main/decaf/lang/exceptions/InterruptedException.h File Reference . . . . .	4659
7.704src/main/decaf/lang/exceptions/InvalidStateException.h File Reference . . . . .	4660
7.705src/main/decaf/lang/exceptions/NoSuchElementException.h File Reference . . . . .	4660
7.706src/main/decaf/lang/exceptions/NullPointerException.h File Reference . . . . .	4661
7.707src/main/decaf/lang/exceptions/NumberFormatException.h File Reference . . . . .	4661
7.708src/main/decaf/lang/exceptions/RuntimeException.h File Reference . . . . .	4662
7.709src/main/decaf/lang/Float.h File Reference . . . . .	4662
7.710src/main/decaf/lang/Integer.h File Reference . . . . .	4663
7.711src/main/decaf/lang/Iterable.h File Reference . . . . .	4663
7.712src/main/decaf/lang/Long.h File Reference . . . . .	4664
7.713src/main/decaf/lang/Math.h File Reference . . . . .	4664
7.714src/main/decaf/lang/Number.h File Reference . . . . .	4664
7.715src/main/decaf/lang/Pointer.h File Reference . . . . .	4665
7.716src/main/decaf/lang/Readable.h File Reference . . . . .	4666
7.717src/main/decaf/lang/Runnable.h File Reference . . . . .	4667
7.718src/main/decaf/lang/Runtime.h File Reference . . . . .	4667
7.719src/main/decaf/lang/Short.h File Reference . . . . .	4667

7.720src/main/decaf/lang/String.h File Reference . . . . .	4668
7.721src/main/decaf/lang/System.h File Reference . . . . .	4668
7.722src/main/decaf/lang/Thread.h File Reference . . . . .	4669
7.723src/main/decaf/lang/ThreadGroup.h File Reference . . . . .	4670
7.724src/main/decaf/lang/Throwable.h File Reference . . . . .	4670
7.725src/main/decaf/net/BindException.h File Reference . . . . .	4671
7.726src/main/decaf/net/ConnectException.h File Reference . . . . .	4671
7.727src/main/decaf/net/HttpRetryException.h File Reference . . . . .	4672
7.728src/main/decaf/net/Inet4Address.h File Reference . . . . .	4672
7.729src/main/decaf/net/Inet6Address.h File Reference . . . . .	4672
7.730src/main/decaf/net/InetAddress.h File Reference . . . . .	4673
7.731src/main/decaf/net/InetSocketAddress.h File Reference . . . . .	4673
7.732src/main/decaf/net/MalformedURLException.h File Reference . . . . .	4674
7.733src/main/decaf/net/NoRouteToHostException.h File Reference . . . . .	4674
7.734src/main/decaf/net/PortUnreachableException.h File Reference . . . . .	4674
7.735src/main/decaf/net/ProtocolException.h File Reference . . . . .	4675
7.736src/main/decaf/net/ServerSocket.h File Reference . . . . .	4675
7.737src/main/decaf/net/ServerSocketFactory.h File Reference . . . . .	4676
7.738src/main/decaf/net/Socket.h File Reference . . . . .	4676
7.739src/main/decaf/net/SocketAddress.h File Reference . . . . .	4677
7.740src/main/decaf/net/SocketError.h File Reference . . . . .	4678
7.741src/main/decaf/net/SocketException.h File Reference . . . . .	4678
7.742src/main/decaf/net/SocketFactory.h File Reference . . . . .	4678
7.743src/main/decaf/net/SocketImpl.h File Reference . . . . .	4679
7.744src/main/decaf/net/SocketImplFactory.h File Reference . . . . .	4680
7.745src/main/decaf/net/SocketOptions.h File Reference . . . . .	4680
7.746src/main/decaf/net/SocketTimeoutException.h File Reference . . . . .	4680
7.747src/main/decaf/net/ssl/SSLContext.h File Reference . . . . .	4681
7.748src/main/decaf/net/ssl/SSLContextSpi.h File Reference . . . . .	4681
7.749src/main/decaf/net/ssl/SSLParameters.h File Reference . . . . .	4682
7.750src/main/decaf/net/ssl/SSLServerSocket.h File Reference . . . . .	4682
7.751src/main/decaf/net/ssl/SSLServerSocketFactory.h File Reference . . . . .	4683
7.752src/main/decaf/net/ssl/SSLSocket.h File Reference . . . . .	4683
7.753src/main/decaf/net/ssl/SSLSocketFactory.h File Reference . . . . .	4684
7.754src/main/decaf/net/UnknownHostException.h File Reference . . . . .	4684
7.755src/main/decaf/net/UnknownServiceException.h File Reference . . . . .	4685
7.756src/main/decaf/net/URI.h File Reference . . . . .	4685
7.757src/main/decaf/net/URISyntaxException.h File Reference . . . . .	4686
7.758src/main/decaf/net/URL.h File Reference . . . . .	4686
7.759src/main/decaf/net/URLDecoder.h File Reference . . . . .	4687
7.760src/main/decaf/net/URLEncoder.h File Reference . . . . .	4687
7.761src/main/decaf/nio/Buffer.h File Reference . . . . .	4687
7.762src/main/decaf/nio/BufferOverflowException.h File Reference . . . . .	4688
7.763src/main/decaf/nio/BufferUnderflowException.h File Reference . . . . .	4688
7.764src/main/decaf/nio/ByteBuffer.h File Reference . . . . .	4689
7.765src/main/decaf/nio/CharBuffer.h File Reference . . . . .	4689
7.766src/main/decaf/nio/DoubleBuffer.h File Reference . . . . .	4690
7.767src/main/decaf/nio/FloatBuffer.h File Reference . . . . .	4691
7.768src/main/decaf/nio/IntBuffer.h File Reference . . . . .	4691
7.769src/main/decaf/nio/InvalidMarkException.h File Reference . . . . .	4692

7.770src/main/decaf/nio/LongBuffer.h File Reference . . . . .	4692
7.771src/main/decaf/nio/ReadOnlyBufferException.h File Reference . . . . .	4693
7.772src/main/decaf/nio/ShortBuffer.h File Reference . . . . .	4693
7.773src/main/decaf/security/auth/x500/X500Principal.h File Reference . . . . .	4694
7.774src/main/decaf/security/cert/Certificate.h File Reference . . . . .	4694
7.775src/main/decaf/security/cert/CertificateEncodingException.h File Reference . . . . .	4695
7.776src/main/decaf/security/cert/CertificateException.h File Reference . . . . .	4696
7.777src/main/decaf/security/cert/CertificateExpiredException.h File Reference . . . . .	4696
7.778src/main/decaf/security/cert/CertificateNotYetValidException.h File Reference . . . . .	4697
7.779src/main/decaf/security/cert/CertificateParsingException.h File Reference . . . . .	4697
7.780src/main/decaf/security/cert/X509Certificate.h File Reference . . . . .	4698
7.781src/main/decaf/security/GeneralSecurityException.h File Reference . . . . .	4698
7.782src/main/decaf/security/InvalidKeyException.h File Reference . . . . .	4698
7.783src/main/decaf/security/Key.h File Reference . . . . .	4699
7.784src/main/decaf/security/KeyException.h File Reference . . . . .	4699
7.785src/main/decaf/security/KeyManagementException.h File Reference . . . . .	4700
7.786src/main/decaf/security/NoSuchAlgorithmException.h File Reference . . . . .	4700
7.787src/main/decaf/security/NoSuchProviderException.h File Reference . . . . .	4701
7.788src/main/decaf/security/Principal.h File Reference . . . . .	4701
7.789src/main/decaf/security/PublicKey.h File Reference . . . . .	4701
7.790src/main/decaf/security/SecureRandom.h File Reference . . . . .	4702
7.791src/main/decaf/security/SecureRandomSpi.h File Reference . . . . .	4702
7.792src/main/decaf/security/SignatureException.h File Reference . . . . .	4703
7.793src/main/decaf/util/AbstractCollection.h File Reference . . . . .	4703
7.794src/main/decaf/util/AbstractList.h File Reference . . . . .	4704
7.795src/main/decaf/util/AbstractMap.h File Reference . . . . .	4705
7.796src/main/decaf/util/AbstractQueue.h File Reference . . . . .	4705
7.797src/main/decaf/util/AbstractSequentialList.h File Reference . . . . .	4706
7.798src/main/decaf/util/AbstractSet.h File Reference . . . . .	4707
7.799src/main/decaf/util/Collection.h File Reference . . . . .	4707
7.800src/main/decaf/util/Comparator.h File Reference . . . . .	4708
7.801src/main/decaf/util/comparators/Less.h File Reference . . . . .	4708
7.802src/main/decaf/util/concurrent/atomic/AtomicBoolean.h File Reference . . . . .	4709
7.803src/main/decaf/util/concurrent/atomic/AtomicInteger.h File Reference . . . . .	4709
7.804src/main/decaf/util/concurrent/atomic/AtomicReferenceCounter.h File Reference . . . . .	4710
7.805src/main/decaf/util/concurrent/atomic/AtomicReference.h File Reference . . . . .	4710
7.806src/main/decaf/util/concurrent/BlockingQueue.h File Reference . . . . .	4711
7.807src/main/decaf/util/concurrent/BrokenBarrierException.h File Reference . . . . .	4712
7.808src/main/decaf/util/concurrent/Callable.h File Reference . . . . .	4712
7.809src/main/decaf/util/concurrent/CancellationException.h File Reference . . . . .	4712
7.810src/main/decaf/util/concurrent/Concurrent.h File Reference . . . . .	4713
7.810.1 Define Documentation . . . . .	4713
7.810.1.1 synchronized . . . . .	4713
7.810.1.2 WAIT_INFINITE . . . . .	4714
7.811src/main/decaf/util/concurrent/ConcurrentMap.h File Reference . . . . .	4714

7.812src/main/decaf/util/concurrent/ConcurrentStlMap.h File Reference . .	4715
7.813src/main/decaf/util/concurrent/CountDownLatch.h File Reference . .	4715
7.814src/main/decaf/util/concurrent/Delayed.h File Reference . . . . .	4716
7.815src/main/decaf/util/concurrent/ExecutionException.h File Reference .	4716
7.816src/main/decaf/util/concurrent/Executor.h File Reference . . . . .	4717
7.817src/main/decaf/util/concurrent/ExecutorService.h File Reference . . .	4717
7.818src/main/decaf/util/concurrent/Future.h File Reference . . . . .	4718
7.819src/main/decaf/util/concurrent/Lock.h File Reference . . . . .	4718
7.820src/main/decaf/util/concurrent/locks/Lock.h File Reference . . . . .	4719
7.821src/main/decaf/util/concurrent/locks/Condition.h File Reference . . .	4719
7.822src/main/decaf/util/concurrent/locks/LockSupport.h File Reference . .	4720
7.823src/main/decaf/util/concurrent/locks/ReadWriteLock.h File Reference	4721
7.824src/main/decaf/util/concurrent/locks/ReentrantLock.h File Reference .	4721
7.825src/main/decaf/util/concurrent/Mutex.h File Reference . . . . .	4722
7.826src/main/decaf/util/concurrent/PooledThread.h File Reference . . . .	4722
7.827src/main/decaf/util/concurrent/PooledThreadListener.h File Reference	4723
7.828src/main/decaf/util/concurrent/RejectedExecutionException.h File Ref- erence . . . . .	4723
7.829src/main/decaf/util/concurrent/RejectedExecutionHandler.h File Refer- ence . . . . .	4724
7.830src/main/decaf/util/concurrent/Semaphore.h File Reference . . . . .	4724
7.831src/main/decaf/util/concurrent/Synchronizable.h File Reference . . . .	4725
7.832src/main/decaf/util/concurrent/SynchronousQueue.h File Reference .	4725
7.833src/main/decaf/util/concurrent/TaskListener.h File Reference . . . . .	4726
7.834src/main/decaf/util/concurrent/ThreadFactory.h File Reference . . . .	4726
7.835src/main/decaf/util/concurrent/ThreadPool.h File Reference . . . . .	4727
7.836src/main/decaf/util/concurrent/TimeoutException.h File Reference . .	4728
7.837src/main/decaf/util/concurrent/TimeUnit.h File Reference . . . . .	4728
7.838src/main/decaf/util/Date.h File Reference . . . . .	4729
7.839src/main/decaf/util/Iterator.h File Reference . . . . .	4729
7.840src/main/decaf/util/List.h File Reference . . . . .	4730
7.841src/main/decaf/util/ListIterator.h File Reference . . . . .	4730
7.842src/main/decaf/util/logging/ConsoleHandler.h File Reference . . . . .	4731
7.843src/main/decaf/util/logging/ErrorHandler.h File Reference . . . . .	4731
7.844src/main/decaf/util/logging/Filter.h File Reference . . . . .	4732
7.845src/main/decaf/util/logging/Formatter.h File Reference . . . . .	4732
7.846src/main/decaf/util/logging/Handler.h File Reference . . . . .	4733
7.847src/main/decaf/util/logging/Level.h File Reference . . . . .	4734
7.848src/main/decaf/util/logging/Logger.h File Reference . . . . .	4734
7.849src/main/decaf/util/logging/LoggerCommon.h File Reference . . . . .	4735
7.850src/main/decaf/util/logging/LoggerDefines.h File Reference . . . . .	4735
7.850.1 Define Documentation . . . . .	4736
7.850.1.1 LOGDECAF_DEBUG . . . . .	4736
7.850.1.2 LOGDECAF_DEBUG_1 . . . . .	4736
7.850.1.3 LOGDECAF_DECLARE . . . . .	4737
7.850.1.4 LOGDECAF_DECLARE_LOCAL . . . . .	4737
7.850.1.5 LOGDECAF_ERROR . . . . .	4737
7.850.1.6 LOGDECAF_FATAL . . . . .	4737
7.850.1.7 LOGDECAF_INFO . . . . .	4737
7.850.1.8 LOGDECAF_INITIALIZE . . . . .	4737

7.850.1.9 LOGDECAF_WARN . . . . .	4737
7.851src/main/decaf/util/logging/LoggerHierarchy.h File Reference . . . . .	4737
7.852src/main/decaf/util/logging/LogManager.h File Reference . . . . .	4737
7.853src/main/decaf/util/logging/LogRecord.h File Reference . . . . .	4738
7.854src/main/decaf/util/logging/LogWriter.h File Reference . . . . .	4739
7.855src/main/decaf/util/logging/MarkBlockLogger.h File Reference . . . . .	4739
7.856src/main/decaf/util/logging/PropertiesChangeListener.h File Reference . . . . .	4740
7.857src/main/decaf/util/logging/SimpleFormatter.h File Reference . . . . .	4740
7.858src/main/decaf/util/logging/SimpleLogger.h File Reference . . . . .	4741
7.859src/main/decaf/util/logging/StreamHandler.h File Reference . . . . .	4741
7.860src/main/decaf/util/logging/XMLFormatter.h File Reference . . . . .	4742
7.861src/main/decaf/util/Map.h File Reference . . . . .	4742
7.862src/main/decaf/util/PriorityQueue.h File Reference . . . . .	4743
7.863src/main/decaf/util/Properties.h File Reference . . . . .	4743
7.864src/main/decaf/util/Random.h File Reference . . . . .	4744
7.865src/main/decaf/util/Set.h File Reference . . . . .	4745
7.866src/main/decaf/util/StlList.h File Reference . . . . .	4745
7.867src/main/decaf/util/StlMap.h File Reference . . . . .	4746
7.868src/main/decaf/util/StlQueue.h File Reference . . . . .	4747
7.869src/main/decaf/util/StlSet.h File Reference . . . . .	4747
7.870src/main/decaf/util/StringTokenizer.h File Reference . . . . .	4748
7.871src/main/decaf/util/Timer.h File Reference . . . . .	4748
7.872src/main/decaf/util/TimerTask.h File Reference . . . . .	4749
7.873src/main/decaf/util/UUID.h File Reference . . . . .	4750
7.874src/main/decaf/util/zip/Adler32.h File Reference . . . . .	4750
7.875src/main/decaf/util/zip/CheckedInputStream.h File Reference . . . . .	4751
7.876src/main/decaf/util/zip/CheckedOutputStream.h File Reference . . . . .	4751
7.877src/main/decaf/util/zip/Checksum.h File Reference . . . . .	4752
7.878src/main/decaf/util/zip/CRC32.h File Reference . . . . .	4752
7.879src/main/decaf/util/zip/DataFormatException.h File Reference . . . . .	4753
7.880src/main/decaf/util/zip/Deflater.h File Reference . . . . .	4753
7.881src/main/decaf/util/zip/DeflaterOutputStream.h File Reference . . . . .	4754
7.882src/main/decaf/util/zip/Inflater.h File Reference . . . . .	4754
7.883src/main/decaf/util/zip/InflaterInputStream.h File Reference . . . . .	4755
7.884src/main/decaf/util/zip/ZipException.h File Reference . . . . .	4756



# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<b>activemq</b> (Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements ) . . . . .	93
<b>activemq::cmsutil</b> . . . . .	94
<b>activemq::commands</b> . . . . .	95
<b>activemq::core</b> . . . . .	96
<b>activemq::core::policies</b> . . . . .	98
<b>activemq::exceptions</b> . . . . .	98
<b>activemq::io</b> . . . . .	98
<b>activemq::library</b> . . . . .	98
<b>activemq::state</b> . . . . .	98
<b>activemq::threads</b> . . . . .	99
<b>activemq::transport</b> . . . . .	99
<b>activemq::transport::correlator</b> . . . . .	100
<b>activemq::transport::failover</b> . . . . .	100
<b>activemq::transport::inactivity</b> . . . . .	101
<b>activemq::transport::logging</b> . . . . .	101
<b>activemq::transport::mock</b> . . . . .	101
<b>activemq::transport::tcp</b> . . . . .	102
<b>activemq::util</b> . . . . .	102
<b>activemq::wireformat</b> . . . . .	103
<b>activemq::wireformat::openwire</b> . . . . .	103
<b>activemq::wireformat::openwire::marshal</b> . . . . .	104
<b>activemq::wireformat::openwire::marshal::v1</b> . . . . .	104
<b>activemq::wireformat::openwire::marshal::v2</b> . . . . .	109
<b>activemq::wireformat::openwire::marshal::v3</b> . . . . .	113
<b>activemq::wireformat::openwire::marshal::v4</b> . . . . .	117
<b>activemq::wireformat::openwire::marshal::v5</b> . . . . .	121
<b>activemq::wireformat::openwire::marshal::v6</b> . . . . .	126
<b>activemq::wireformat::openwire::utils</b> . . . . .	130

<b>activemq::wireformat::stomp</b> . . . . .	130
<b>cms</b> (Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements ) . . . . .	131
<b>decaf</b> (Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements ) . . . . .	134
<b>decaf::internal</b> . . . . .	135
<b>decaf::internal::io</b> . . . . .	135
<b>decaf::internal::net</b> . . . . .	135
<b>decaf::internal::net::ssl</b> . . . . .	136
<b>decaf::internal::net::ssl::openssl</b> . . . . .	136
<b>decaf::internal::net::tcp</b> . . . . .	137
<b>decaf::internal::nio</b> . . . . .	138
<b>decaf::internal::security</b> . . . . .	138
<b>decaf::internal::util</b> . . . . .	138
<b>decaf::internal::util::concurrent</b> . . . . .	139
<b>decaf::io</b> . . . . .	139
<b>decaf::lang</b> . . . . .	141
<b>decaf::lang::exceptions</b> . . . . .	144
<b>decaf::net</b> . . . . .	145
<b>decaf::net::ssl</b> . . . . .	146
<b>decaf::nio</b> . . . . .	147
<b>decaf::security</b> . . . . .	147
<b>decaf::security::auth</b> . . . . .	148
<b>decaf::security::auth::x500</b> . . . . .	148
<b>decaf::security::cert</b> . . . . .	148
<b>decaf::util</b> . . . . .	149
<b>decaf::util::comparators</b> . . . . .	151
<b>decaf::util::concurrent</b> . . . . .	151
<b>decaf::util::concurrent::atomic</b> . . . . .	153
<b>decaf::util::concurrent::locks</b> . . . . .	153
<b>decaf::util::logging</b> . . . . .	154
<b>decaf::util::zip</b> . . . . .	156
<b>std</b> . . . . .	157

## Chapter 2

# Data Structure Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

activemq::core::ActiveMQAckHandler . . . . .	183
activemq::core::ActiveMQConstants . . . . .	297
activemq::library::ActiveMQCPP . . . . .	310
activemq::core::ActiveMQTransactionContext . . . . .	727
decaf::lang::Appendable . . . . .	733
decaf::io::Writer . . . . .	4133
decaf::io::OutputStreamWriter . . . . .	2999
decaf::nio::CharBuffer . . . . .	1148
decaf::internal::nio::CharArrayBuffer . . . . .	1135
decaf::internal::AprPool . . . . .	735
decaf::lang::ArrayPointer< T, REFCOUNTER > . . . . .	736
decaf::util::concurrent::atomic::AtomicBoolean . . . . .	745
decaf::util::concurrent::atomic::AtomicRefCounter . . . . .	754
decaf::util::concurrent::atomic::AtomicReference< T > . . . . .	756
binary_function . . . . .	842
decaf::util::Comparator< ArrayPointer< T, R > > . . . . .	1251
decaf::lang::ArrayPointerComparator< T, R > . . . . .	744
decaf::util::Comparator< E > . . . . .	1251
decaf::util::comparators::Less< E > . . . . .	2399
decaf::util::Comparator< Pointer< T, R > > . . . . .	1251
decaf::lang::PointerComparator< T, R > . . . . .	3040
decaf::util::Comparator< T > . . . . .	1251
std::less< decaf::lang::ArrayPointer< T > > . . . . .	2401
std::less< decaf::lang::Pointer< T > > . . . . .	2402
activemq::wireformat::openwire::utils::BooleanStream . . . . .	863
decaf::nio::Buffer . . . . .	936
decaf::nio::ByteBuffer . . . . .	1049
decaf::internal::nio::ByteArrayBuffer . . . . .	1004

decaf::nio::CharBuffer . . . . .	1148
decaf::nio::DoubleBuffer . . . . .	1865
decaf::internal::nio::DoubleArrayBuffer . . . . .	1853
decaf::nio::FloatBuffer . . . . .	1985
decaf::internal::nio::FloatArrayBuffer . . . . .	1974
decaf::nio::IntBuffer . . . . .	2130
decaf::internal::nio::IntArrayBuffer . . . . .	2119
decaf::nio::LongBuffer . . . . .	2522
decaf::internal::nio::LongArrayBuffer . . . . .	2511
decaf::nio::ShortBuffer . . . . .	3560
decaf::internal::nio::ShortArrayBuffer . . . . .	3548
decaf::internal::nio::BufferFactory . . . . .	951
decaf::internal::util::ByteArrayAdapter . . . . .	979
decaf::util::concurrent::Callable< V > . . . . .	1108
decaf::security::cert::Certificate . . . . .	1112
decaf::security::cert::X509Certificate . . . . .	4141
decaf::lang::CharSequence . . . . .	1167
decaf::lang::String . . . . .	3775
decaf::nio::CharBuffer . . . . .	1148
decaf::util::zip::Checksum . . . . .	1174
decaf::util::zip::Adler32 . . . . .	730
decaf::util::zip::CRC32 . . . . .	1568
cms::Closeable . . . . .	1179
activemq::commands::ActiveMQTempDestination . . . . .	578
activemq::commands::ActiveMQTempQueue . . . . .	606
activemq::commands::ActiveMQTempTopic . . . . .	636
cms::Connection . . . . .	1296
activemq::core::ActiveMQConnection . . . . .	260
cms::MessageConsumer . . . . .	2674
activemq::cmsutil::CachedConsumer . . . . .	1097
activemq::core::ActiveMQConsumer . . . . .	300
cms::MessageProducer . . . . .	2809
activemq::cmsutil::CachedProducer . . . . .	1101
activemq::core::ActiveMQProducer . . . . .	466
cms::QueueBrowser . . . . .	3243
activemq::core::ActiveMQQueueBrowser . . . . .	483
cms::Session . . . . .	3460
activemq::cmsutil::PooledSession . . . . .	3041
activemq::core::ActiveMQSession . . . . .	512
decaf::io::Closeable . . . . .	1180
activemq::transport::Transport . . . . .	3996
activemq::transport::CompositeTransport . . . . .	1259
activemq::transport::failover::FailoverTransport . . . . .	1930
activemq::transport::IOTransport . . . . .	2213
activemq::transport::mock::MockTransport . . . . .	2854
activemq::transport::TransportFilter . . . . .	4004
activemq::transport::correlator::ResponseCorrelator . . . . .	3384

activemq::transport::inactivity::InactivityMonitor . . . . .	2065
activemq::transport::logging::LoggingTransport . . . . .	2477
activemq::transport::tcp::TcpTransport . . . . .	3865
activemq::transport::tcp::SslTransport . . . . .	3682
activemq::wireformat::WireFormatNegotiator . . . . .	4129
activemq::wireformat::openwire::OpenWireFormatNegotiator . . . . .	2985
decaf::io::InputStream . . . . .	2105
decaf::internal::io::StandardInputStream . . . . .	3688
decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream . . . . .	2966
decaf::internal::net::tcp::TcpSocketInputStream . . . . .	3860
decaf::io::BlockingByteArrayInputStream . . . . .	845
decaf::io::ByteArrayInputStream . . . . .	1038
decaf::io::FilterInputStream . . . . .	1950
activemq::io::LoggingInputStream . . . . .	2474
decaf::io::BufferedInputStream . . . . .	943
decaf::io::DataInputStream . . . . .	1612
decaf::io::PushbackInputStream . . . . .	3231
decaf::util::zip::CheckedInputStream . . . . .	1169
decaf::util::zip::InflaterInputStream . . . . .	2097
decaf::io::OutputStream . . . . .	2991
decaf::internal::io::StandardErrorOutputStream . . . . .	3686
decaf::internal::io::StandardOutputStream . . . . .	3689
decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream . . . . .	2969
decaf::internal::net::tcp::TcpSocketOutputStream . . . . .	3863
decaf::io::ByteArrayOutputStream . . . . .	1046
decaf::io::FilterOutputStream . . . . .	1957
activemq::io::LoggingOutputStream . . . . .	2476
decaf::io::BufferedOutputStream . . . . .	949
decaf::io::DataOutputStream . . . . .	1628
decaf::util::zip::CheckedOutputStream . . . . .	1172
decaf::util::zip::DeflaterOutputStream . . . . .	1769
decaf::io::Reader . . . . .	3254
decaf::io::InputStreamReader . . . . .	2116
decaf::io::Writer . . . . .	4133
decaf::net::Socket . . . . .	3607
decaf::net::ssl::SSLSocket . . . . .	3670
decaf::internal::net::ssl::openssl::OpenSSLSocket . . . . .	2940
decaf::util::logging::Handler . . . . .	2042
decaf::util::logging::StreamHandler . . . . .	3756
decaf::util::logging::ConsoleHandler . . . . .	1439
activemq::cmsutil::CmsAccessor . . . . .	1183
activemq::cmsutil::CmsDestinationAccessor . . . . .	1187
activemq::cmsutil::CmsTemplate . . . . .	1201
cms::CMSException . . . . .	1190
cms::CMSSecurityException . . . . .	1199
cms::IllegalStateException . . . . .	2059
cms::InvalidClientIdException . . . . .	2199
cms::InvalidDestinationException . . . . .	2200

cms::InvalidSelectorException . . . . .	2206
cms::MessageEOFException . . . . .	2747
cms::MessageFormatException . . . . .	2749
cms::MessageNotReadableException . . . . .	2807
cms::MessageNotWriteableException . . . . .	2808
cms::UnsupportedOperationException . . . . .	4030
activemq::util::CMSExceptionSupport . . . . .	1194
cms::CMSProperties . . . . .	1195
activemq::util::ActiveMQProperties . . . . .	474
code . . . . .	1215
activemq::state::CommandVisitor . . . . .	1233
activemq::state::CommandVisitorAdapter . . . . .	1242
activemq::state::ConnectionStateTracker . . . . .	1432
decaf::lang::Comparable< T > . . . . .	1248
decaf::lang::Comparable< bool > . . . . .	1248
decaf::lang::Boolean . . . . .	856
decaf::lang::Comparable< Boolean > . . . . .	1248
decaf::lang::Boolean . . . . .	856
decaf::lang::Comparable< BrokerId > . . . . .	1248
activemq::commands::BrokerId . . . . .	874
decaf::lang::Comparable< Byte > . . . . .	1248
decaf::lang::Byte . . . . .	969
decaf::lang::Comparable< ByteBuffer > . . . . .	1248
decaf::nio::ByteBuffer . . . . .	1049
decaf::lang::Comparable< char > . . . . .	1248
decaf::lang::Character . . . . .	1126
decaf::lang::Comparable< Character > . . . . .	1248
decaf::lang::Character . . . . .	1126
decaf::lang::Comparable< CharBuffer > . . . . .	1248
decaf::nio::CharBuffer . . . . .	1148
decaf::lang::Comparable< ConnectionId > . . . . .	1248
activemq::commands::ConnectionId . . . . .	1365
decaf::lang::Comparable< ConsumerId > . . . . .	1248
activemq::commands::ConsumerId . . . . .	1472
decaf::lang::Comparable< Date > . . . . .	1248
decaf::util::Date . . . . .	1719
decaf::lang::Comparable< Delayed > . . . . .	1248
decaf::util::concurrent::Delayed . . . . .	1774
decaf::lang::Comparable< Double > . . . . .	1248
decaf::lang::Double . . . . .	1841
decaf::lang::Comparable< double > . . . . .	1248
decaf::lang::Double . . . . .	1841
decaf::lang::Comparable< DoubleBuffer > . . . . .	1248
decaf::nio::DoubleBuffer . . . . .	1865

decaf::lang::Comparable< float > . . . . .	1248
decaf::lang::Float . . . . .	1961
decaf::lang::Comparable< Float > . . . . .	1248
decaf::lang::Float . . . . .	1961
decaf::lang::Comparable< FloatBuffer > . . . . .	1248
decaf::nio::FloatBuffer . . . . .	1985
decaf::lang::Comparable< int > . . . . .	1248
decaf::lang::Integer . . . . .	2143
decaf::lang::Comparable< IntBuffer > . . . . .	1248
decaf::nio::IntBuffer . . . . .	2130
decaf::lang::Comparable< Integer > . . . . .	1248
decaf::lang::Integer . . . . .	2143
decaf::lang::Comparable< Level > . . . . .	1248
decaf::util::logging::Level . . . . .	2403
decaf::lang::Comparable< LocalTransactionId > . . . . .	1248
activemq::commands::LocalTransactionId . . . . .	2420
decaf::lang::Comparable< Long > . . . . .	1248
decaf::lang::Long . . . . .	2495
decaf::lang::Comparable< long long > . . . . .	1248
decaf::lang::Long . . . . .	2495
decaf::lang::Comparable< LongBuffer > . . . . .	1248
decaf::nio::LongBuffer . . . . .	2522
decaf::lang::Comparable< MessageId > . . . . .	1248
activemq::commands::MessageId . . . . .	2750
decaf::lang::Comparable< ProducerId > . . . . .	1248
activemq::commands::ProducerId . . . . .	3157
decaf::lang::Comparable< SessionId > . . . . .	1248
activemq::commands::SessionId . . . . .	3476
decaf::lang::Comparable< Short > . . . . .	1248
decaf::lang::Short . . . . .	3538
decaf::lang::Comparable< short > . . . . .	1248
decaf::lang::Short . . . . .	3538
decaf::lang::Comparable< ShortBuffer > . . . . .	1248
decaf::nio::ShortBuffer . . . . .	3560
decaf::lang::Comparable< TimeUnit > . . . . .	1248
decaf::util::concurrent::TimeUnit . . . . .	3919
decaf::lang::Comparable< TransactionId > . . . . .	1248
activemq::commands::TransactionId . . . . .	3932
activemq::commands::LocalTransactionId . . . . .	2420
activemq::commands::XATransactionId . . . . .	4143
decaf::lang::Comparable< unsigned char > . . . . .	1248
decaf::lang::Byte . . . . .	969
decaf::lang::Comparable< URI > . . . . .	1248

decaf::net::URI	4031
decaf::lang::Comparable< UUID >	1248
decaf::util::UUID	4080
decaf::lang::Comparable< XATransactionId >	1248
activemq::commands::XATransactionId	4143
activemq::util::CompositeData	1253
decaf::util::concurrent::locks::Condition	1282
decaf::util::concurrent::ConditionHandle	1290
decaf::internal::util::concurrent::ConditionImpl	1291
cms::ConnectionFactory	1361
activemq::core::ActiveMQConnectionFactory	281
cms::ConnectionMetaData	1425
activemq::core::ActiveMQConnectionMetaData	293
activemq::state::ConnectionState	1429
activemq::state::ConsumerState	1535
decaf::util::concurrent::CountDownLatch	1565
ct_data_s	1571
decaf::io::DataInput	1603
decaf::io::DataOutput	1622
activemq::wireformat::openwire::marshal::DataStreamMarshaller	1660
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller	814
activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller	328
activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller	491
activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller	587
activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller	615
activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller	649
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller	710
activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller	786
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller	919
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller	1315
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	1349
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller	1412
activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller	1459
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	1522
activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller	1552
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	1797
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	2019
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	2359
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller	2665
activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller	2707
activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller	2737
activemq::wireformat::openwire::marshal::v1::MessageMarshaller	2798
activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller	194
activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller	239
activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller	369
activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller	397
activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller	445
activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller	557



```
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller680
activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller2845
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller3150
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller3199
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller3300
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller3318
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller3351
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller 3408
    activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller1587
    activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller1656
    activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller1919
    activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller2180
    activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller3517
    activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller3584
    activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller3968
activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller . . 886
activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller 1381
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller 1489
activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller1831
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller2248
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller2277
activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller 2300
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller2331
activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller . 2775
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller2901
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller3028
    activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller2395
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller . 3181
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller . . 3501
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller3791
activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller3939
    activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller2446
    activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller4160
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller4121
activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller340
    activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller504
    activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller599
        activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller628
        activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller658
    activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller722
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller807
    activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller932
    activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller1328
    activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller1336
    activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller1400
    activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller1447
    activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller1510
    activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller1539
    activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller1784
```

activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller2006  
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller2342  
 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller2653  
 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller2690  
 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller2725  
 activemq::wireformat::openwire::marshal::v2::MessageMarshaller 2789  
     activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller202  
     activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller256  
     activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller381  
     activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller410  
     activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller457  
     activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller570  
     activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller693  
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller2828  
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller3128  
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller3194  
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller3288  
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller3326  
 activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller3355  
 activemq::wireformat::openwire::marshal::v2::ResponseMarshaller 3393  
     activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller1574  
     activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller1643  
     activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller1902  
     activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller2167  
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller3525  
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller3580  
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller3985  
 activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller . . 898  
 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller 1368  
 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller 1477  
 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller1819  
 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller2232  
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller2261  
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller 2283  
 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller2315  
 activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller . 2755  
 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller2881  
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller3010  
     activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller2382  
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller . 3161  
 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller . . 3481  
 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller3807  
 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller3943  
     activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller2429  
     activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller4151  
 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller4113  
 activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller324  
     activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller487  
     activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller582

```
    activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller611
    activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller641
    activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller701
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller771
    activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller910
    activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller1307
    activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller1340
    activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller1404
    activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller1451
    activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller1514
    activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller1544
    activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller1789
    activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller2010
    activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller2347
    activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller2657
    activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller2695
    activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller2729
    activemq::wireformat::openwire::marshal::v3::MessageMarshaller 2785
        activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller189
        activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller235
        activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller364
        activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller393
        activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller440
        activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller553
        activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller671
    activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller2837
    activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller3137
    activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller3207
    activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller3296
    activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller3322
    activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller3360
    activemq::wireformat::openwire::marshal::v3::ResponseMarshaller 3403
        activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller1579
        activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller1647
        activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller1906
        activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller2171
    activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller3521
    activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller3593
    activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller3972
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller 878
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller 1373
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller 1481
activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller1823
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller2240
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller2265
activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller 2288
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller2319
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller 2767
activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller2893
```

activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller3019  
     activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller2378  
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller . 3169  
 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller . . 3497  
 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller3786  
 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller3947  
     activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller2433  
     activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller4164  
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller4125  
 activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller332  
     activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller495  
     activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller591  
         activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller619  
         activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller645  
     activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller705  
 activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller778  
     activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller915  
     activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller1311  
     activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller1344  
     activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller1408  
     activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller1455  
     activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller1518  
     activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller1548  
     activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller1793  
     activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller2014  
     activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller2351  
     activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller2661  
     activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller2703  
     activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller2733  
     activemq::wireformat::openwire::marshal::v4::MessageMarshaller 2793  
         activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller198  
         activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller243  
         activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller373  
         activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller401  
         activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller449  
         activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller561  
         activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller676  
     activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller2841  
     activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller3133  
     activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller3190  
     activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller3309  
     activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller3339  
     activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller3347  
     activemq::wireformat::openwire::marshal::v4::ResponseMarshaller 3388  
         activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller1583  
         activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller1652  
         activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller1915  
         activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller2175  
     activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller3530

```
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller3597
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller3981
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller . . 882
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller 1377
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller 1485
activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller1827
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller2244
activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller2273
activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller 2296
activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller2327
activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller . 2759
activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller2897
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller3023
    activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller2391
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller . 3165
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller . . 3485
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller3799
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller3951
    activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller2442
    activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller4155
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller4117
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller336
    activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller499
    activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller595
        activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller624
        activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller654
    activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller714
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller793
    activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller923
    activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller1319
    activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller1353
    activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller1417
    activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller1464
    activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller1527
    activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller1556
    activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller1806
    activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller2023
    activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller2355
    activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller2670
    activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller2699
    activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller2742
    activemq::wireformat::openwire::marshal::v5::MessageMarshaller 2780
        activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller206
        activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller247
        activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller377
        activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller405
        activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller453
        activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller566
        activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller684
```

activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller2833  
 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller3141  
 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller3203  
 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller3305  
 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller3335  
 activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller3368  
 activemq::wireformat::openwire::marshal::v5::ResponseMarshaller 3398  
     activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller1592  
     activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller1635  
     activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller1910  
     activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller2184  
 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller3513  
 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller3589  
 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller3964  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller . . 890  
 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller 1385  
 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller 1493  
 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller1835  
 activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller2236  
 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller2256  
 activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller 2304  
 activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller2323  
 activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller . 2763  
 activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller2889  
 activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller3014  
     activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller2386  
 activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller . 3173  
 activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller . . 3493  
 activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller3795  
 activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller3935  
     activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller2437  
     activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller4168  
 activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller4104  
 activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller344  
     activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller508  
     activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller603  
         activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller632  
         activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller662  
     activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller718  
 activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller800  
     activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller927  
     activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller1324  
     activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller1357  
     activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller1421  
     activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller1468  
     activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller1531  
     activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller1561  
     activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller1801  
     activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller2002

activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller	2338
activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller	2648
activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller	2712
activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller	2720
activemq::wireformat::openwire::marshal::v6::MessageMarshaller	2802
activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller	211
activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller	252
activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller	386
activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller	414
activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller	462
activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller	574
activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller	688
activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller	2850
activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller	3145
activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller	3211
activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller	3292
activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller	3331
activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller	3364
activemq::wireformat::openwire::marshal::v6::ResponseMarshaller	3413
activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller	1596
activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller	1639
activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller	1898
activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller	2162
activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller	3508
activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller	3576
activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller	3977
activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller	894
activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller	1389
activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller	1497
activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller	1815
activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller	2228
activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller	2269
activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller	2292
activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller	2311
activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller	2771
activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller	2885
activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller	3005
activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller	2374
activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller	3177
activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller	3489
activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller	3803
activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller	3955
activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller	2425
activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller	4147
activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller	4109
decaf::internal::net::ssl::DefaultSSLContext	1743
decaf::util::zip::Deflater	1759
cms::DeliveryMode	1775
cms::Destination	1776

cms::Queue . . . . .	3238
activemq::commands::ActiveMQQueue . . . . .	479
cms::TemporaryQueue . . . . .	3871
activemq::commands::ActiveMQTempQueue . . . . .	606
cms::TemporaryTopic . . . . .	3873
activemq::commands::ActiveMQTempTopic . . . . .	636
cms::Topic . . . . .	3930
activemq::commands::ActiveMQTopic . . . . .	697
activemq::commands::ActiveMQDestination::DestinationFilter . . . . .	1779
activemq::cmsutil::DestinationResolver . . . . .	1810
activemq::cmsutil::DynamicDestinationResolver . . . . .	1878
activemq::core::DispatchData . . . . .	1839
activemq::core::Dispatcher . . . . .	1840
activemq::core::ActiveMQConsumer . . . . .	300
activemq::core::ActiveMQSession . . . . .	512
decaf::lang::DYNAMIC_CAST_TOKEN . . . . .	1878
decaf::util::Map< K, V, COMPARATOR >::Entry . . . . .	1880
decaf::util::logging::ErrorManager . . . . .	1884
cms::ExceptionListener . . . . .	1893
decaf::util::concurrent::Executor . . . . .	1926
decaf::util::concurrent::ExecutorService . . . . .	1928
decaf::io::FileDescriptor . . . . .	1947
decaf::internal::net::SocketFileDescriptor . . . . .	3634
decaf::util::logging::Filter . . . . .	1949
decaf::io::Flushable . . . . .	1998
decaf::io::OutputStream . . . . .	2991
decaf::io::Writer . . . . .	4133
decaf::util::logging::Formatter . . . . .	2027
decaf::util::logging::SimpleFormatter . . . . .	3604
decaf::util::logging::XMLFormatter . . . . .	4172
decaf::util::concurrent::Future< V > . . . . .	2029
activemq::transport::correlator::FutureResponse . . . . .	2032
gz_header_s . . . . .	2038
gz_state . . . . .	2039
decaf::internal::util::HexStringParser . . . . .	2046
activemq::wireformat::openwire::utils::HexTable . . . . .	2048
activemq::util::IdGenerator . . . . .	2052
decaf::net::InetAddress . . . . .	2077
decaf::net::Inet4Address . . . . .	2072
decaf::net::Inet6Address . . . . .	2076
inflate_state . . . . .	2085
decaf::util::zip::Inflater . . . . .	2088
internal_state . . . . .	2188
decaf::lang::Iterable< E > . . . . .	2220
decaf::util::Collection< E > . . . . .	1216
decaf::util::AbstractCollection< E > . . . . .	159
decaf::util::List< E > . . . . .	2409



decaf::util::AbstractList< E > . . . . .	173
decaf::util::AbstractSequentialList< E > . . . . .	179
decaf::util::StlList< E > . . . . .	3694
decaf::util::Queue< E > . . . . .	3239
decaf::util::AbstractQueue< E > . . . . .	175
decaf::util::concurrent::BlockingQueue< E > . . . . .	848
decaf::util::concurrent::SynchronousQueue< E > . . . . .	3827
decaf::util::PriorityQueue< E > . . . . .	3116
decaf::util::Set< E > . . . . .	3538
decaf::util::AbstractSet< E > . . . . .	180
decaf::util::StlSet< E > . . . . .	3731
decaf::lang::Iterable< PrimitiveValueNode > . . . . .	2220
decaf::util::Collection< PrimitiveValueNode > . . . . .	1216
decaf::util::AbstractCollection< PrimitiveValueNode > . . . . .	159
decaf::util::List< PrimitiveValueNode > . . . . .	2409
decaf::util::StlList< PrimitiveValueNode > . . . . .	3694
activemq::util::PrimitiveList . . . . .	3067
decaf::util::Iterator< T > . . . . .	2222
decaf::util::Iterator< E > . . . . .	2222
decaf::util::ListIterator< E > . . . . .	2417
decaf::security::Key . . . . .	2364
decaf::security::PublicKey . . . . .	3231
decaf::util::concurrent::Lock . . . . .	2450
decaf::util::concurrent::locks::Lock . . . . .	2452
decaf::util::concurrent::locks::ReentrantLock . . . . .	3273
decaf::util::concurrent::locks::LockSupport . . . . .	2458
decaf::util::logging::Logger . . . . .	2461
decaf::util::logging::LoggerHierarchy . . . . .	2474
decaf::util::logging::LogManager . . . . .	2480
decaf::util::logging::LogRecord . . . . .	2487
decaf::util::logging::LogWriter . . . . .	2493
activemq::util::LongSequenceGenerator . . . . .	2535
decaf::util::logging::MarkBlockLogger . . . . .	2563
activemq::wireformat::MarshalAware . . . . .	2564
activemq::commands::DataStructure . . . . .	1713
activemq::commands::BaseDataStructure . . . . .	837
activemq::commands::ActiveMQDestination . . . . .	312
activemq::commands::ActiveMQQueue . . . . .	479
activemq::commands::ActiveMQTempDestination . . . . .	578
activemq::commands::ActiveMQTopic . . . . .	697
activemq::commands::BooleanExpression . . . . .	861
activemq::commands::BrokerId . . . . .	874
activemq::commands::Command . . . . .	1227
activemq::commands::BaseCommand . . . . .	764
activemq::commands::BrokerError . . . . .	868
activemq::commands::BrokerInfo . . . . .	902
activemq::commands::ConnectionControl . . . . .	1301

activemq::commands::ConnectionError . . . . .	1332
activemq::commands::ConnectionInfo . . . . .	1393
activemq::commands::ConsumerControl . . . . .	1441
activemq::commands::ConsumerInfo . . . . .	1501
activemq::commands::ControlCommand . . . . .	1536
activemq::commands::DestinationInfo . . . . .	1779
activemq::commands::FlushCommand . . . . .	1999
activemq::commands::KeepAliveInfo . . . . .	2335
activemq::commands::Message . . . . .	2596
activemq::commands::ActiveMQMessageTemplate< T > . . . . .	418
activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage > . . . . .	418
activemq::commands::ActiveMQBytesMessage . . . . .	215
activemq::commands::ActiveMQMessageTemplate< cms::MapMessage > . . . . .	418
activemq::commands::ActiveMQMapMessage . . . . .	350
activemq::commands::ActiveMQMessageTemplate< cms::Message > . . . . .	418
activemq::commands::ActiveMQBlobMessage . . . . .	184
activemq::commands::ActiveMQMessage . . . . .	390
activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage > . . . . .	418
activemq::commands::ActiveMQObjectMessage . . . . .	438
activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage > . . . . .	418
activemq::commands::ActiveMQStreamMessage . . . . .	535
activemq::commands::ActiveMQMessageTemplate< cms::TextMessage > . . . . .	418
activemq::commands::ActiveMQTextMessage . . . . .	666
activemq::commands::MessageAck . . . . .	2643
activemq::commands::MessageDispatch . . . . .	2678
activemq::commands::MessageDispatchNotification . . . . .	2716
activemq::commands::MessagePull . . . . .	2824
activemq::commands::ProducerAck . . . . .	3124
activemq::commands::ProducerInfo . . . . .	3185
activemq::commands::RemoveInfo . . . . .	3284
activemq::commands::RemoveSubscriptionInfo . . . . .	3313
activemq::commands::ReplayCommand . . . . .	3343
activemq::commands::Response . . . . .	3379
activemq::commands::DataArrayResponse . . . . .	1572
activemq::commands::DataResponse . . . . .	1632
activemq::commands::ExceptionResponse . . . . .	1894
activemq::commands::IntegerResponse . . . . .	2160
activemq::state::Tracked . . . . .	3931
activemq::commands::SessionInfo . . . . .	3505
activemq::commands::ShutdownInfo . . . . .	3573
activemq::commands::TransactionInfo . . . . .	3959
activemq::commands::WireFormatInfo . . . . .	4093
activemq::commands::ConnectionId . . . . .	1365
activemq::commands::ConsumerId . . . . .	1472

activemq::commands::DiscoveryEvent . . . . .	1812
activemq::commands::JournalQueueAck . . . . .	2224
activemq::commands::JournalTopicAck . . . . .	2252
activemq::commands::JournalTrace . . . . .	2281
activemq::commands::JournalTransaction . . . . .	2308
activemq::commands::MessageId . . . . .	2750
activemq::commands::NetworkBridgeFilter . . . . .	2877
activemq::commands::PartialCommand . . . . .	3002
activemq::commands::LastPartialCommand . . . . .	2371
activemq::commands::ProducerId . . . . .	3157
activemq::commands::SessionId . . . . .	3476
activemq::commands::SubscriptionInfo . . . . .	3782
activemq::commands::TransactionId . . . . .	3932
activemq::wireformat::openwire::marshal::v6::MarshallerFactory . . . . .	2567
activemq::wireformat::openwire::marshal::v3::MarshallerFactory . . . . .	2567
activemq::wireformat::openwire::marshal::v4::MarshallerFactory . . . . .	2568
activemq::wireformat::openwire::marshal::v5::MarshallerFactory . . . . .	2569
activemq::wireformat::openwire::marshal::v1::MarshallerFactory . . . . .	2570
activemq::wireformat::openwire::marshal::v2::MarshallerFactory . . . . .	2570
activemq::util::MarshallingSupport . . . . .	2571
decaf::lang::Math . . . . .	2575
cms::Message . . . . .	2614
activemq::commands::ActiveMQMessageTemplate< cms::Message > . . . . .	418
cms::BytesMessage . . . . .	1079
activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage > . . . . .	418
cms::MapMessage . . . . .	2551
activemq::commands::ActiveMQMessageTemplate< cms::MapMessage > . . . . .	418
cms::ObjectMessage . . . . .	2924
activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage > . . . . .	418
cms::StreamMessage . . . . .	3760
activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage > . . . . .	418
cms::TextMessage . . . . .	3874
activemq::commands::ActiveMQMessageTemplate< cms::TextMessage > . . . . .	418
activemq::cmsutil::MessageCreator . . . . .	2677
cms::MessageEnumeration . . . . .	2746
activemq::core::ActiveMQQueueBrowser . . . . .	483
cms::MessageListener . . . . .	2779
activemq::wireformat::openwire::utils::MessagePropertyInterceptor . . . . .	2817
decaf::util::concurrent::MutexHandle . . . . .	2871
decaf::internal::util::concurrent::MutexImpl . . . . .	2872
decaf::internal::net::Network . . . . .	2874
decaf::lang::Number . . . . .	2918
decaf::lang::Byte . . . . .	969

decaf::lang::Character . . . . .	1126
decaf::lang::Double . . . . .	1841
decaf::lang::Float . . . . .	1961
decaf::lang::Integer . . . . .	2143
decaf::lang::Long . . . . .	2495
decaf::lang::Short . . . . .	3538
decaf::util::concurrent::atomic::AtomicInteger . . . . .	748
decaf::internal::net::ssl::openssl::OpenSSLParameters . . . . .	2927
decaf::lang::Pointer< T, REFCOUNTER > . . . . .	3032
decaf::util::concurrent::PooledThreadListener . . . . .	3058
decaf::util::concurrent::ThreadPool . . . . .	3888
activemq::core::PrefetchPolicy . . . . .	3062
activemq::core::policies::DefaultPrefetchPolicy . . . . .	1726
activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller . . . . .	3090
activemq::util::PrimitiveValueNode::PrimitiveValue . . . . .	3097
activemq::util::PrimitiveValueConverter . . . . .	3098
activemq::util::PrimitiveValueNode . . . . .	3099
decaf::security::Principal . . . . .	3114
decaf::security::auth::x500::X500Principal . . . . .	4140
activemq::cmsutil::ProducerCallback . . . . .	3154
activemq::cmsutil::CmsTemplate::SendExecutor . . . . .	3445
activemq::state::ProducerState . . . . .	3216
decaf::util::Properties . . . . .	3216
decaf::util::logging::PropertiesChangeListener . . . . .	3227
decaf::util::Random . . . . .	3245
decaf::security::SecureRandom . . . . .	3424
decaf::lang::Readable . . . . .	3251
decaf::io::Reader . . . . .	3254
decaf::util::concurrent::locks::ReadWriteLock . . . . .	3263
activemq::core::RedeliveryPolicy . . . . .	3267
activemq::core::policies::DefaultRedeliveryPolicy . . . . .	1730
decaf::util::concurrent::RejectedExecutionHandler . . . . .	3283
decaf::internal::util::Resource . . . . .	3374
decaf::internal::util::GenericResource< T > . . . . .	2037
decaf::internal::util::ResourceLifecycleManager . . . . .	3375
activemq::cmsutil::ResourceLifecycleManager . . . . .	3376
activemq::transport::mock::ResponseBuilder . . . . .	3382
activemq::wireformat::openwire::OpenWireResponseBuilder . . . . .	2989
decaf::lang::Runnable . . . . .	3418
activemq::threads::CompositeTaskRunner . . . . .	1256
activemq::threads::DedicatedTaskRunner . . . . .	1724
activemq::transport::IOTransport . . . . .	2213
decaf::lang::Thread . . . . .	3876
activemq::transport::mock::InternalCommandListener . . . . .	2192
decaf::util::concurrent::PooledThread . . . . .	3056
decaf::util::TimerTask . . . . .	3914

activemq::transport::inactivity::ReadChecker . . . . .	3253
activemq::transport::inactivity::WriteChecker . . . . .	4132
decaf::lang::Runtime . . . . .	3419
decaf::internal::DecafRuntime . . . . .	1723
decaf::security::SecureRandomSpi . . . . .	3432
decaf::internal::security::SecureRandomImpl . . . . .	3429
decaf::internal::security::SecureRandomImpl . . . . .	3429
decaf::util::concurrent::Semaphore . . . . .	3434
decaf::net::ServerSocket . . . . .	3447
decaf::net::ssl::SSLServerSocket . . . . .	3662
decaf::internal::net::ssl::openssl::OpenSSLServerSocket . . . . .	2929
decaf::net::ServerSocketFactory . . . . .	3457
decaf::internal::net::DefaultServerSocketFactory . . . . .	1735
decaf::net::ssl::SSLServerSocketFactory . . . . .	3668
decaf::internal::net::ssl::DefaultSSLServerSocketFactory . . . . .	1744
decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory . . . . .	2935
activemq::cmsutil::SessionCallback . . . . .	3475
activemq::cmsutil::CmsTemplate::ProducerExecutor . . . . .	3155
activemq::cmsutil::CmsTemplate::ResolveProducerExecutor . . . . .	3372
activemq::cmsutil::CmsTemplate::ReceiveExecutor . . . . .	3265
activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor . . . . .	3373
activemq::cmsutil::SessionPool . . . . .	3534
activemq::state::SessionState . . . . .	3536
decaf::util::logging::SimpleLogger . . . . .	3605
decaf::net::SocketAddress . . . . .	3626
decaf::net::InetSocketAddress . . . . .	2085
decaf::net::SocketError . . . . .	3626
decaf::net::SocketFactory . . . . .	3629
decaf::internal::net::DefaultSocketFactory . . . . .	1738
decaf::net::ssl::SSLSocketFactory . . . . .	3679
decaf::internal::net::ssl::DefaultSSLSocketFactory . . . . .	1749
decaf::internal::net::ssl::openssl::OpenSSLSocketFactory . . . . .	2959
decaf::net::SocketImplFactory . . . . .	3644
decaf::net::SocketOptions . . . . .	3645
decaf::net::SocketImpl . . . . .	3635
decaf::internal::net::tcp::TcpSocket . . . . .	3850
decaf::net::ssl::SSLContext . . . . .	3653
decaf::net::ssl::SSLContextSpi . . . . .	3655
decaf::internal::net::ssl::openssl::OpenSSLContextSpi . . . . .	2924
decaf::net::ssl::SSLParameters . . . . .	3658
activemq::commands::BrokerError::StackTraceElement . . . . .	3685
cms::Startable . . . . .	3691
cms::Connection . . . . .	1296
decaf::lang::STATIC_CAST_TOKEN . . . . .	3692
activemq::core::ActiveMQConstants::StaticInitializer . . . . .	3693
activemq::wireformat::stomp::StompCommandConstants . . . . .	3738

activemq::wireformat::stomp::StompFrame	3741
activemq::wireformat::stomp::StompHelper	3746
cms::Stoppable	3755
cms::Connection	1296
decaf::util::StringTokenizer	3779
decaf::util::concurrent::Synchronizable	3811
activemq::core::MessageDispatchChannel	2683
decaf::util::Collection< PrimitiveValueNode >	1216
decaf::internal::util::concurrent::SynchronizableImpl	3822
decaf::io::InputStream	2105
decaf::io::OutputStream	2991
decaf::util::Collection< E >	1216
decaf::util::concurrent::Mutex	2866
decaf::util::Map< K, V, COMPARATOR >	2538
decaf::util::AbstractMap< K, V, COMPARATOR >	174
decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >	1260
decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR	
>	1265
decaf::util::StlMap< K, V, COMPARATOR >	3708
decaf::util::StlQueue< T >	3722
decaf::util::Map< std::string, PrimitiveValueNode, std::less< std::string	
> >	2538
decaf::util::StlMap< std::string, PrimitiveValueNode >	3708
activemq::util::PrimitiveMap	3079
activemq::core::Synchronization	3826
decaf::lang::System	3838
activemq::threads::Task	3846
activemq::core::ActiveMQSessionExecutor	532
activemq::threads::CompositeTask	1255
activemq::transport::failover::BackupTransportPool	761
activemq::transport::failover::CloseTransportsTask	1182
activemq::transport::failover::FailoverTransport	1930
activemq::threads::CompositeTaskRunner	1256
decaf::util::concurrent::TaskListener	3847
activemq::threads::TaskRunner	3849
activemq::threads::CompositeTaskRunner	1256
activemq::threads::DedicatedTaskRunner	1724
decaf::util::concurrent::ThreadFactory	3887
decaf::lang::ThreadGroup	3888
decaf::lang::Throwable	3895
decaf::lang::Exception	1886
activemq::exceptions::ActiveMQException	348
activemq::exceptions::BrokerException	873
decaf::io::IOException	2210
decaf::io::EOFException	1881
decaf::io::InterruptedIOException	2196
decaf::net::SocketTimeoutException	3650

decaf::io::UnsupportedEncodingException . . . . .	4025
decaf::io::UTFDataFormatException . . . . .	4078
decaf::net::HttpRetryException . . . . .	2049
decaf::net::MalformedURLException . . . . .	2536
decaf::net::ProtocolException . . . . .	3228
decaf::net::SocketException . . . . .	3627
decaf::internal::net::ssl::openssl::OpenSSLSocketException . . . . .	2955
decaf::net::BindException . . . . .	842
decaf::net::ConnectException . . . . .	1293
decaf::net::NoRouteToHostException . . . . .	2905
decaf::net::PortUnreachableException . . . . .	3060
decaf::net::UnknownHostException . . . . .	4019
decaf::net::UnknownServiceException . . . . .	4022
decaf::util::zip::ZipException . . . . .	4175
decaf::lang::exceptions::ClassCastException . . . . .	1177
decaf::lang::exceptions::IllegalArgumentException . . . . .	2054
decaf::lang::exceptions::IllegalMonitorStateException . . . . .	2056
decaf::lang::exceptions::IllegalStateException . . . . .	2060
decaf::nio::InvalidMarkException . . . . .	2204
decaf::lang::exceptions::IllegalThreadStateException . . . . .	2063
decaf::lang::exceptions::IndexOutOfBoundsException . . . . .	2069
decaf::lang::exceptions::InterruptedException . . . . .	2193
decaf::lang::exceptions::InvalidStateException . . . . .	2207
decaf::lang::exceptions::NoSuchElementException . . . . .	2910
decaf::lang::exceptions::NullPointerException . . . . .	2915
decaf::lang::exceptions::NumberFormatException . . . . .	2921
decaf::lang::exceptions::RuntimeException . . . . .	3421
decaf::lang::exceptions::UnsupportedOperationException . . . . .	4028
decaf::nio::ReadOnlyBufferException . . . . .	3260
decaf::net::URISyntaxException . . . . .	4060
decaf::nio::BufferOverflowException . . . . .	964
decaf::nio::BufferUnderflowException . . . . .	967
decaf::security::GeneralSecurityException . . . . .	2034
decaf::security::cert::CertificateException . . . . .	1118
decaf::security::cert::CertificateEncodingException . . . . .	1116
decaf::security::cert::CertificateExpiredException . . . . .	1120
decaf::security::cert::CertificateNotYetValidException . . . . .	1122
decaf::security::cert::CertificateParsingException . . . . .	1124
decaf::security::KeyException . . . . .	2366
decaf::security::InvalidKeyException . . . . .	2201
decaf::security::KeyManagementException . . . . .	2368
decaf::security::NoSuchAlgorithmException . . . . .	2907
decaf::security::NoSuchProviderException . . . . .	2913
decaf::security::SignatureException . . . . .	3601
decaf::util::concurrent::BrokenBarrierException . . . . .	866
decaf::util::concurrent::CancellationException . . . . .	1110
decaf::util::concurrent::ExecutionException . . . . .	1923
decaf::util::concurrent::RejectedExecutionException . . . . .	3280
decaf::util::concurrent::TimeoutException . . . . .	3899

decaf::util::zip::DataFormatException	1600
decaf::util::Timer	3901
decaf::internal::util::TimerTaskHeap	3916
activemq::state::TransactionState	3989
decaf::internal::util::concurrent::Transferer< E >	3991
decaf::internal::util::concurrent::TransferQueue< E >	3992
decaf::internal::util::concurrent::TransferStack< E >	3994
activemq::transport::TransportFactory	4003
activemq::transport::AbstractTransportFactory	182
activemq::transport::failover::FailoverTransportFactory	1942
activemq::transport::mock::MockTransportFactory	2864
activemq::transport::tcp::TcpTransportFactory	3869
activemq::transport::tcp::SslTransportFactory	3684
activemq::transport::TransportListener	4013
activemq::core::ActiveMQConnection	260
activemq::transport::DefaultTransportListener	1757
activemq::transport::failover::BackupTransport	759
activemq::transport::mock::InternalCommandListener	2192
activemq::transport::failover::FailoverTransportListener	1945
activemq::transport::TransportFilter	4004
activemq::transport::TransportRegistry	4015
tree_desc_s	4018
decaf::lang::Thread::UncaughtExceptionHandler	4018
decaf::internal::net::URIEncoderDecoder	4044
decaf::internal::net::URIHelper	4047
activemq::transport::failover::URIPool	4054
activemq::util::URISupport	4057
decaf::internal::net::URIType	4063
decaf::net::URL	4072
decaf::net::URLDecoder	4074
decaf::net::URLEncoder	4074
activemq::util::Usage	4075
activemq::util::MemoryUsage	2592
activemq::wireformat::WireFormat	4088
activemq::wireformat::openwire::OpenWireFormat	2971
activemq::wireformat::stomp::StompWireFormat	3751
activemq::wireformat::WireFormatFactory	4092
activemq::wireformat::openwire::OpenWireFormatFactory	2984
activemq::wireformat::stomp::StompWireFormatFactory	3754
activemq::wireformat::WireFormatRegistry	4129
z_stream_s	4174



## Chapter 3

# Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

<b>decaf::util::AbstractCollection</b> < <b>E</b> > (This class provides a skeletal implementation of the <b>Collection</b> (p. 1216) interface, to minimize the effort required to implement this interface ) . . . . .	159
<b>decaf::util::AbstractList</b> < <b>E</b> > (This class provides a skeletal implementation of the <b>List</b> (p. 2409) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array) ) . . . . .	173
<b>decaf::util::AbstractMap</b> < <b>K</b> , <b>V</b> , <b>COMPARATOR</b> > (This class provides a skeletal implementation of the <b>Map</b> (p. 2538) interface, to minimize the effort required to implement this interface ) . . . . .	174
<b>decaf::util::AbstractQueue</b> < <b>E</b> > (This class provides skeletal implementations of some <b>Queue</b> (p. 3239) operations ) . . . . .	175
<b>decaf::util::AbstractSequentialList</b> < <b>E</b> > (This class provides a skeletal implementation of the <b>List</b> (p. 2409) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list) ) . . . . .	179
<b>decaf::util::AbstractSet</b> < <b>E</b> > (This class provides a skeletal implementation of the <b>Set</b> (p. 3538) interface to minimize the effort required to implement this interface ) . . . . .	180
<b>activemq::transport::AbstractTransportFactory</b> (Abstract implementation of the <b>TransportFactory</b> (p. 4003) interface, providing the base functionality that's common to most of the <b>TransportFactory</b> (p. 4003) instances ) . . . . .	182
<b>activemq::core::ActiveMQAckHandler</b> (Interface class that is used to give CMS Messages an interface to Ack themselves with ) . . . . .	183
<b>activemq::commands::ActiveMQBlobMessage</b> . . . . .	184
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBlobMessageMarshaller</b> (p. 189) ) . . . . .	189

<b>activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBlobMessageMarshaller</b> (p. 194) ) . . . . .	194
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBlobMessageMarshaller</b> (p. 198) ) . . . . .	198
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBlobMessageMarshaller</b> (p. 202) ) . . . . .	202
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBlobMessageMarshaller</b> (p. 206) ) . . . . .	206
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBlobMessageMarshaller</b> (p. 211) ) . . . . .	211
<b>activemq::commands::ActiveMQBytesMessage</b> . . . . .	215
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBytesMessageMarshaller</b> (p. 235) ) . . . . .	235
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBytesMessageMarshaller</b> (p. 239) ) . . . . .	239
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBytesMessageMarshaller</b> (p. 243) ) . . . . .	243
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBytesMessageMarshaller</b> (p. 247) ) . . . . .	247
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBytesMessageMarshaller</b> (p. 252) ) . . . . .	252
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBytesMessageMarshaller</b> (p. 256) ) . . . . .	256
<b>activemq::core::ActiveMQConnection</b> (Concrete connection used for all connectors to the ActiveMQ broker) . . . . .	260
<b>activemq::core::ActiveMQConnectionFactory</b> . . . . .	281
<b>activemq::core::ActiveMQConnectionMetaData</b> (This class houses all the various settings and information that is used by an instance of an <b>ActiveMQConnection</b> (p. 260) class) . . . . .	293
<b>activemq::core::ActiveMQConstants</b> (Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values) . . . . .	297
<b>activemq::core::ActiveMQConsumer</b> . . . . .	300
<b>activemq::library::ActiveMQCPP</b> . . . . .	310
<b>activemq::commands::ActiveMQDestination</b> . . . . .	312
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQDestinationMarshaller</b> (p. 324) ) . . . . .	324

<b>activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQDestinationMarshaller</b> (p. 328) ) . . . . .	328
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQDestinationMarshaller</b> (p. 332) ) . . . . .	332
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQDestinationMarshaller</b> (p. 336) ) . . . . .	336
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQDestinationMarshaller</b> (p. 340) ) . . . . .	340
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQDestinationMarshaller</b> (p. 344) ) . . . . .	344
<b>activemq::exceptions::ActiveMQException</b> . . . . .	348
<b>activemq::commands::ActiveMQMapMessage</b> . . . . .	350
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMapMessageMarshaller</b> (p. 364) ) . . . . .	364
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMapMessageMarshaller</b> (p. 369) ) . . . . .	369
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMapMessageMarshaller</b> (p. 373) ) . . . . .	373
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMapMessageMarshaller</b> (p. 377) ) . . . . .	377
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMapMessageMarshaller</b> (p. 381) ) . . . . .	381
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMapMessageMarshaller</b> (p. 386) ) . . . . .	386
<b>activemq::commands::ActiveMQMessage</b> . . . . .	390
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMessageMarshaller</b> (p. 393) ) . . . . .	393
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMessageMarshaller</b> (p. 397) ) . . . . .	397
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMessageMarshaller</b> (p. 401) ) . . . . .	401
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMessageMarshaller</b> (p. 405) ) . . . . .	405

<b>activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMessageMarshaller</b> (p. 410) ) . . . . .	410
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMessageMarshaller</b> (p. 414) ) . . . . .	414
<b>activemq::commands::ActiveMQMessageTemplate&lt; T &gt;</b> . . . . .	418
<b>activemq::commands::ActiveMQObjectMessage</b> . . . . .	438
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQObjectMessageMarshaller</b> (p. 440) ) . . . . .	440
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQObjectMessageMarshaller</b> (p. 445) ) . . . . .	445
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQObjectMessageMarshaller</b> (p. 449) ) . . . . .	449
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQObjectMessageMarshaller</b> (p. 453) ) . . . . .	453
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQObjectMessageMarshaller</b> (p. 457) ) . . . . .	457
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQObjectMessageMarshaller</b> (p. 462) ) . . . . .	462
<b>activemq::core::ActiveMQProducer</b> . . . . .	466
<b>activemq::util::ActiveMQProperties</b> (Implementation of the CMSProperties interface that delegates to a <b>decaf::util::Properties</b> (p. 3216) object ) . . . . .	474
<b>activemq::commands::ActiveMQQueue</b> . . . . .	479
<b>activemq::core::ActiveMQQueueBrowser</b> . . . . .	483
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQQueueMarshaller</b> (p. 487) ) . . . . .	487
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQQueueMarshaller</b> (p. 491) ) . . . . .	491
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQQueueMarshaller</b> (p. 495) ) . . . . .	495
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQQueueMarshaller</b> (p. 499) ) . . . . .	499
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQQueueMarshaller</b> (p. 504) ) . . . . .	504
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQQueueMarshaller</b> (p. 508) ) . . . . .	508

<b>activemq::core::ActiveMQSession</b> . . . . .	512
<b>activemq::core::ActiveMQSessionExecutor</b> (Delegate dispatcher for a single session ) . . . . .	532
<b>activemq::commands::ActiveMQStreamMessage</b> . . . . .	535
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQStreamMessageMarshaller</b> (p. 553) ) . . . . .	553
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQStreamMessageMarshaller</b> (p. 557) ) . . . . .	557
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQStreamMessageMarshaller</b> (p. 561) ) . . . . .	561
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQStreamMessageMarshaller</b> (p. 566) ) . . . . .	566
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQStreamMessageMarshaller</b> (p. 570) ) . . . . .	570
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQStreamMessageMarshaller</b> (p. 574) ) . . . . .	574
<b>activemq::commands::ActiveMQTempDestination</b> . . . . .	578
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempDestinationMarshaller</b> (p. 582) ) . . . . .	582
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempDestinationMarshaller</b> (p. 587) ) . . . . .	587
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempDestinationMarshaller</b> (p. 591) ) . . . . .	591
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempDestinationMarshaller</b> (p. 595) ) . . . . .	595
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempDestinationMarshaller</b> (p. 599) ) . . . . .	599
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempDestinationMarshaller</b> (p. 603) ) . . . . .	603
<b>activemq::commands::ActiveMQTempQueue</b> . . . . .	606
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempQueueMarshaller</b> (p. 611) ) . . . . .	611
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempQueueMarshaller</b> (p. 615) ) . . . . .	615

<b>activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempQueue-</b> <b>Marshaller</b> (p. 619) ) . . . . .	619
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempQueue-</b> <b>Marshaller</b> (p. 624) ) . . . . .	624
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempQueue-</b> <b>Marshaller</b> (p. 628) ) . . . . .	628
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempQueue-</b> <b>Marshaller</b> (p. 632) ) . . . . .	632
<b>activemq::commands::ActiveMQTempTopic</b> . . . . .	636
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempTopic-</b> <b>Marshaller</b> (p. 641) ) . . . . .	641
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempTopic-</b> <b>Marshaller</b> (p. 645) ) . . . . .	645
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempTopic-</b> <b>Marshaller</b> (p. 649) ) . . . . .	649
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempTopic-</b> <b>Marshaller</b> (p. 654) ) . . . . .	654
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempTopic-</b> <b>Marshaller</b> (p. 658) ) . . . . .	658
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempTopic-</b> <b>Marshaller</b> (p. 662) ) . . . . .	662
<b>activemq::commands::ActiveMQTextMessage</b> . . . . .	666
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTextMes-</b> <b>sageMarshaller</b> (p. 671) ) . . . . .	671
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTextMes-</b> <b>sageMarshaller</b> (p. 676) ) . . . . .	676
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTextMes-</b> <b>sageMarshaller</b> (p. 680) ) . . . . .	680
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTextMes-</b> <b>sageMarshaller</b> (p. 684) ) . . . . .	684
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTextMes-</b> <b>sageMarshaller</b> (p. 688) ) . . . . .	688
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTextMes-</b> <b>sageMarshaller</b> (p. 693) ) . . . . .	693

<b>activemq::commands::ActiveMQTopic</b>	697
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTopicMarshaller</b> (p. 701) )	701
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTopicMarshaller</b> (p. 705) )	705
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTopicMarshaller</b> (p. 710) )	710
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTopicMarshaller</b> (p. 714) )	714
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTopicMarshaller</b> (p. 718) )	718
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTopicMarshaller</b> (p. 722) )	722
<b>activemq::core::ActiveMQTransactionContext</b> (Transaction Management class, hold messages that are to be redelivered upon a request to roll-back )	727
<b>decaf::util::zip::Adler32</b> (Clas that can be used to compute an Adler-32 Checksum (p. 1174) for a data stream )	730
<b>decaf::lang::Appendable</b> (An object to which char sequences and values can be appended )	733
<b>decaf::internal::AprPool</b> (Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made )	735
<b>decaf::lang::ArrayPointer&lt; T, REFCOUNTER &gt;</b> (Decaf's implementation of a Smart <b>Pointer</b> (p. 3032) that is a template on a Type and is <b>Thread</b> (p. 3876) Safe if the default Reference Counter is used )	736
<b>decaf::lang::ArrayPointerComparator&lt; T, R &gt;</b> (This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this <b>ArrayPointer</b> (p. 736) )	744
<b>decaf::util::concurrent::atomic::AtomicBoolean</b> (A boolean value that may be updated atomically )	745
<b>decaf::util::concurrent::atomic::AtomicInteger</b> (An int value that may be updated atomically )	748
<b>decaf::util::concurrent::atomic::AtomicRefCounter</b>	754
<b>decaf::util::concurrent::atomic::AtomicReference&lt; T &gt;</b> (An Pointer reference that may be updated atomically )	756
<b>activemq::transport::failover::BackupTransport</b>	759
<b>activemq::transport::failover::BackupTransportPool</b>	761
<b>activemq::commands::BaseCommand</b>	764
<b>activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>BaseCommandMarshaller</b> (p. 771) )	771

<b>activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>BaseCommandMarshaller</b> (p. 778) ) . . . . .	778
<b>activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>BaseCommandMarshaller</b> (p. 786) ) . . . . .	786
<b>activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>BaseCommandMarshaller</b> (p. 793) ) . . . . .	793
<b>activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>BaseCommandMarshaller</b> (p. 800) ) . . . . .	800
<b>activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>BaseCommandMarshaller</b> (p. 807) ) . . . . .	807
<b>activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller</b> (Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol ) . . . . .	814
<b>activemq::commands::BaseDataStructure</b> . . . . .	837
<b>binary_function</b> . . . . .	842
<b>decaf::net::BindException</b> . . . . .	842
<b>decaf::io::BlockingByteArrayInputStream</b> (This is a blocking version of a byte buffer stream ) . . . . .	845
<b>decaf::util::concurrent::BlockingQueue&lt; E &gt;</b> (A <b>decaf::util::Queue</b> (p. 3239) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element ) . . . . .	848
<b>decaf::lang::Boolean</b> . . . . .	856
<b>activemq::commands::BooleanExpression</b> . . . . .	861
<b>activemq::wireformat::openwire::utils::BooleanStream</b> (Manages the writing and reading of boolean data streams to and from a data source such as a <b>DataInputStream</b> or <b>DataOutputStream</b> ) . . . . .	863
<b>decaf::util::concurrent::BrokenBarrierException</b> . . . . .	866
<b>activemq::commands::BrokerError</b> (This class represents an Exception sent from the Broker ) . . . . .	868
<b>activemq::exceptions::BrokerException</b> . . . . .	873
<b>activemq::commands::BrokerId</b> . . . . .	874
<b>activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerIdMarshaller</b> (p. 878) ) . . . . .	878
<b>activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerIdMarshaller</b> (p. 882) ) . . . . .	882
<b>activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerIdMarshaller</b> (p. 886) ) . . . . .	886
<b>activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerIdMarshaller</b> (p. 890) ) . . . . .	890



<b>activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerIdMarshaller</b> (p. 894) ) . . . . .	894
<b>activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerIdMarshaller</b> (p. 898) ) . . . . .	898
<b>activemq::commands::BrokerInfo</b> . . . . .	902
<b>activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerInfoMarshaller</b> (p. 910) ) . . . . .	910
<b>activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerInfoMarshaller</b> (p. 915) ) . . . . .	915
<b>activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerInfoMarshaller</b> (p. 919) ) . . . . .	919
<b>activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerInfoMarshaller</b> (p. 923) ) . . . . .	923
<b>activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerInfoMarshaller</b> (p. 927) ) . . . . .	927
<b>activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerInfoMarshaller</b> (p. 932) ) . . . . .	932
<b>decaf::nio::Buffer</b> (A container for data of a specific primitive type ) . . . .	936
<b>decaf::io::BufferedInputStream</b> (A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream ) .	943
<b>decaf::io::BufferedOutputStream</b> (Wrapper around another output stream that buffers output before writing to the target output stream ) . . . .	949
<b>decaf::internal::nio::BufferFactory</b> (Factory class used by static methods in the <b>decaf::nio</b> (p. 147) package to create the various default version of the NIO interfaces ) . . . . .	951
<b>decaf::nio::BufferOverflowException</b> . . . . .	964
<b>decaf::nio::BufferUnderflowException</b> . . . . .	967
<b>decaf::lang::Byte</b> . . . . .	969
<b>decaf::internal::util::ByteArrayAdapter</b> (This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data ) . . . . .	979
<b>decaf::internal::nio::ByteBuffer</b> (This class defines six categories of operations upon byte buffers: ) . . . . .	1004
<b>decaf::io::ByteArrayInputStream</b> (A <b>ByteArrayInputStream</b> (p. 1038) contains an internal buffer that contains bytes that may be read from the stream ) . . . . .	1038
<b>decaf::io::ByteArrayOutputStream</b> . . . . .	1046
<b>decaf::nio::ByteBuffer</b> (This class defines six categories of operations upon byte buffers: ) . . . . .	1049
<b>cms::BytesMessage</b> (A <b>BytesMessage</b> (p. 1079) object is used to send a message containing a stream of unsigned bytes ) . . . . .	1079

<b>activemq::cmsutil::CachedConsumer</b> (A cached message consumer contained within a pooled session ) . . . . .	1097
<b>activemq::cmsutil::CachedProducer</b> (A cached message producer contained within a pooled session ) . . . . .	1101
<b>decaf::util::concurrent::Callable&lt; V &gt;</b> (A task that returns a result and may throw an exception ) . . . . .	1108
<b>decaf::util::concurrent::CancellationException</b> . . . . .	1110
<b>decaf::security::cert::Certificate</b> (Base interface for all identity certificates )	1112
<b>decaf::security::cert::CertificateEncodingException</b> . . . . .	1116
<b>decaf::security::cert::CertificateException</b> . . . . .	1118
<b>decaf::security::cert::CertificateExpiredException</b> . . . . .	1120
<b>decaf::security::cert::CertificateNotYetValidException</b> . . . . .	1122
<b>decaf::security::cert::CertificateParsingException</b> . . . . .	1124
<b>decaf::lang::Character</b> . . . . .	1126
<b>decaf::internal::nio::CharArrayBuffer</b> . . . . .	1135
<b>decaf::nio::CharBuffer</b> (This class defines four categories of operations upon character buffers: ) . . . . .	1148
<b>decaf::lang::CharSequence</b> (A <b>CharSequence</b> (p. 1167) is a readable sequence of char values ) . . . . .	1167
<b>decaf::util::zip::CheckedInputStream</b> (An implementation of a <b>FilterInputStream</b> that will maintain a <b>Checksum</b> (p. 1174) of the bytes read, the <b>Checksum</b> (p. 1174) can then be used to verify the integrity of the input stream ) . . . . .	1169
<b>decaf::util::zip::CheckedOutputStream</b> (An implementation of a <b>FilterOutputStream</b> that will maintain a <b>Checksum</b> (p. 1174) of the bytes written, the <b>Checksum</b> (p. 1174) can then be used to verify the integrity of the output stream ) . . . . .	1172
<b>decaf::util::zip::Checksum</b> (An interface used to represent <b>Checksum</b> (p. 1174) values in the Zip package ) . . . . .	1174
<b>decaf::lang::exceptions::ClassCastException</b> . . . . .	1177
<b>cms::Closeable</b> (Interface for a class that implements the close method ) . . .	1179
<b>decaf::io::Closeable</b> (Interface for a class that implements the close method )	1180
<b>activemq::transport::failover::CloseTransportsTask</b> . . . . .	1182
<b>activemq::cmsutil::CmsAccessor</b> (Base class for <b>activemq::cmsutil::CmsTemplate</b> (p. 1201) and other CMS-accessing gateway helpers, defining common properties such as the CMS <b>cms.ConnectionFactory</b> (p. 1361) to operate on ) . . . . .	1183
<b>activemq::cmsutil::CmsDestinationAccessor</b> (Extends the <b>CmsAccessor</b> (p. 1183) to add support for resolving destination names ) . . . . .	1187
<b>cms::CMSException</b> (CMS API Exception that is the base for all exceptions thrown from CMS classes ) . . . . .	1190
<b>activemq::util::CMSExceptionSupport</b> . . . . .	1194
<b>cms::CMSProperties</b> (Interface for a Java-like properties object ) . . . . .	1195
<b>cms::CMSSecurityException</b> (This exception must be thrown when a provider rejects a user name/password submitted by a client ) . . . . .	1199
<b>activemq::cmsutil::CmsTemplate</b> ( <b>CmsTemplate</b> (p. 1201) simplifies performing synchronous CMS operations ) . . . . .	1201
<b>code</b> . . . . .	1215
<b>decaf::util::Collection&lt; E &gt;</b> (The root interface in the collection hierarchy )	1216
<b>activemq::commands::Command</b> . . . . .	1227

<b>activemq::state::CommandVisitor</b> (Interface for an Object that can visit the various Command Objects that are sent from and to this client ) .	1233
<b>activemq::state::CommandVisitorAdapter</b> (Default Implementation of a <b>CommandVisitor</b> (p. 1233) that returns NULL for all calls ) . . . .	1242
<b>decaf::lang::Comparable&lt; T &gt;</b> (This interface imposes a total ordering on the objects of each class that implements it ) . . . . .	1248
<b>decaf::util::Comparator&lt; T &gt;</b> (A comparison function, which imposes a total ordering on some collection of objects ) . . . . .	1251
<b>activemq::util::CompositeData</b> (Represents a Composite URI ) . . . . .	1253
<b>activemq::threads::CompositeTask</b> (Represents a single task that can be part of a set of Tasks that are contained in a <b>CompositeTaskRunner</b> (p. 1256) ) . . . . .	1255
<b>activemq::threads::CompositeTaskRunner</b> (A <b>Task</b> (p. 3846) Runner that can contain one or more <b>CompositeTasks</b> that are each checked for pending work and run if any is present in the order that the tasks were added ) . . . . .	1256
<b>activemq::transport::CompositeTransport</b> (A <b>Composite Transport</b> (p. 3996) is a <b>Transport</b> (p. 3996) implementation that is composed of several <b>Transports</b> ) . . . . .	1259
<b>decaf::util::concurrent::ConcurrentMap&lt; K, V, COMPARATOR &gt;</b> (Interface for a <b>Map</b> (p. 2538) type that provides additional atomic <b>putIfAbsent</b> , <b>remove</b> , and <b>replace</b> methods alongside the already available <b>Map</b> (p. 2538) interface ) . . . . .	1260
<b>decaf::util::concurrent::ConcurrentStdMap&lt; K, V, COMPARATOR &gt;</b> ( <b>Map</b> (p. 2538) template that wraps around a <b>std::map</b> to provide a more user-friendly interface and to provide common functions that do not exist in <b>std::map</b> ) . . . . .	1265
<b>decaf::util::concurrent::locks::Condition</b> ( <b>Condition</b> (p. 1282) factors out the <b>Mutex</b> (p. 2866) monitor methods ( <b>wait</b> , <b>notify</b> and <b>notifyAll</b> ) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary <b>Lock</b> (p. 2452) implementations ) . . . . .	1282
<b>decaf::util::concurrent::ConditionHandle</b> . . . . .	1290
<b>decaf::internal::util::concurrent::ConditionImpl</b> . . . . .	1291
<b>decaf::net::ConnectException</b> . . . . .	1293
<b>cms::Connection</b> (The client's connection to its provider ) . . . . .	1296
<b>activemq::commands::ConnectionControl</b> . . . . .	1301
<b>activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionControlMarshaller</b> (p. 1307) ) . . . . .	1307
<b>activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionControlMarshaller</b> (p. 1311) ) . . . . .	1311
<b>activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionControlMarshaller</b> (p. 1315) ) . . . . .	1315
<b>activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionControlMarshaller</b> (p. 1319) ) . . . . .	1319

<b>activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionControl-</b> <b>Marshaller</b> (p. 1324) ) . . . . .	1324
<b>activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionControl-</b> <b>Marshaller</b> (p. 1328) ) . . . . .	1328
<b>activemq::commands::ConnectionError</b> . . . . .	1332
<b>activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionError-</b> <b>Marshaller</b> (p. 1336) ) . . . . .	1336
<b>activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionError-</b> <b>Marshaller</b> (p. 1340) ) . . . . .	1340
<b>activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionError-</b> <b>Marshaller</b> (p. 1344) ) . . . . .	1344
<b>activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionError-</b> <b>Marshaller</b> (p. 1349) ) . . . . .	1349
<b>activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionError-</b> <b>Marshaller</b> (p. 1353) ) . . . . .	1353
<b>activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionError-</b> <b>Marshaller</b> (p. 1357) ) . . . . .	1357
<b>cms::ConnectionFactory</b> (Defines the interface for a factory that creates connection objects, the <b>Connection</b> (p. 1296) objects returned im- plement the CMS <b>Connection</b> (p. 1296) interface and hide the CMS Provider specific implementation details behind that interface ) . . .	1361
<b>activemq::commands::ConnectionId</b> . . . . .	1365
<b>activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionIdMar-</b> <b>shaller</b> (p. 1368) ) . . . . .	1368
<b>activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionIdMar-</b> <b>shaller</b> (p. 1373) ) . . . . .	1373
<b>activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionIdMar-</b> <b>shaller</b> (p. 1377) ) . . . . .	1377
<b>activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionIdMar-</b> <b>shaller</b> (p. 1381) ) . . . . .	1381
<b>activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionIdMar-</b> <b>shaller</b> (p. 1385) ) . . . . .	1385
<b>activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionIdMar-</b> <b>shaller</b> (p. 1389) ) . . . . .	1389
<b>activemq::commands::ConnectionInfo</b> . . . . .	1393

<b>activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionInfoMarshaller</b> (p. 1400) ) . . . . .	1400
<b>activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionInfoMarshaller</b> (p. 1404) ) . . . . .	1404
<b>activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionInfoMarshaller</b> (p. 1408) ) . . . . .	1408
<b>activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionInfoMarshaller</b> (p. 1412) ) . . . . .	1412
<b>activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionInfoMarshaller</b> (p. 1417) ) . . . . .	1417
<b>activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionInfoMarshaller</b> (p. 1421) ) . . . . .	1421
<b>cms::ConnectionMetaData</b> (A <b>ConnectionMetaData</b> (p. 1425) object provides information describing the <b>Connection</b> (p. 1296) object ) . . .	1425
<b>activemq::state::ConnectionState</b> . . . . .	1429
<b>activemq::state::ConnectionStateTracker</b> . . . . .	1432
<b>decaf::util::logging::ConsoleHandler</b> (This <b>Handler</b> (p. 2042) publishes log records to System.err ) . . . . .	1439
<b>activemq::commands::ConsumerControl</b> . . . . .	1441
<b>activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerControlMarshaller</b> (p. 1447) ) . . . . .	1447
<b>activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerControlMarshaller</b> (p. 1451) ) . . . . .	1451
<b>activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerControlMarshaller</b> (p. 1455) ) . . . . .	1455
<b>activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerControlMarshaller</b> (p. 1459) ) . . . . .	1459
<b>activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerControlMarshaller</b> (p. 1464) ) . . . . .	1464
<b>activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerControlMarshaller</b> (p. 1468) ) . . . . .	1468
<b>activemq::commands::ConsumerId</b> . . . . .	1472
<b>activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerIdMarshaller</b> (p. 1477) ) . . . . .	1477
<b>activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerIdMarshaller</b> (p. 1481) ) . . . . .	1481

<b>activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerIdMarshaller</b> (p. 1485) ) . . . . .	1485
<b>activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerIdMarshaller</b> (p. 1489) ) . . . . .	1489
<b>activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerIdMarshaller</b> (p. 1493) ) . . . . .	1493
<b>activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerIdMarshaller</b> (p. 1497) ) . . . . .	1497
<b>activemq::commands::ConsumerInfo</b> . . . . .	1501
<b>activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerInfoMarshaller</b> (p. 1510) ) . . . . .	1510
<b>activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerInfoMarshaller</b> (p. 1514) ) . . . . .	1514
<b>activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerInfoMarshaller</b> (p. 1518) ) . . . . .	1518
<b>activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerInfoMarshaller</b> (p. 1522) ) . . . . .	1522
<b>activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerInfoMarshaller</b> (p. 1527) ) . . . . .	1527
<b>activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerInfoMarshaller</b> (p. 1531) ) . . . . .	1531
<b>activemq::state::ConsumerState</b> . . . . .	1535
<b>activemq::commands::ControlCommand</b> . . . . .	1536
<b>activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ControlCommandMarshaller</b> (p. 1539) ) . . . . .	1539
<b>activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ControlCommandMarshaller</b> (p. 1544) ) . . . . .	1544
<b>activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ControlCommandMarshaller</b> (p. 1548) ) . . . . .	1548
<b>activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ControlCommandMarshaller</b> (p. 1552) ) . . . . .	1552
<b>activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ControlCommandMarshaller</b> (p. 1556) ) . . . . .	1556

<b>activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ControlCommandMarshaller</b> (p. 1561) ) . . . . .	1561
<b>decaf::util::concurrent::CountDownLatch</b> . . . . .	1565
<b>decaf::util::zip::CRC32</b> (Class that can be used to compute a CRC-32 checksum for a data stream ) . . . . .	1568
<b>ct_data_s</b> . . . . .	1571
<b>activemq::commands::DataArrayResponse</b> . . . . .	1572
<b>activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>DataArrayResponseMarshaller</b> (p. 1574) ) . . . . .	1574
<b>activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>DataArrayResponseMarshaller</b> (p. 1579) ) . . . . .	1579
<b>activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>DataArrayResponseMarshaller</b> (p. 1583) ) . . . . .	1583
<b>activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>DataArrayResponseMarshaller</b> (p. 1587) ) . . . . .	1587
<b>activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>DataArrayResponseMarshaller</b> (p. 1592) ) . . . . .	1592
<b>activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>DataArrayResponseMarshaller</b> (p. 1596) ) . . . . .	1596
<b>decaf::util::zip::DataFormatException</b> . . . . .	1600
<b>decaf::io::DataInput</b> (The <b>DataInput</b> (p. 1603) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types ) . . . . .	1603
<b>decaf::io::DataInputStream</b> (A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way ) . . . . .	1612
<b>decaf::io::DataOutput</b> (The <b>DataOutput</b> (p. 1622) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream ) . . . . .	1622
<b>decaf::io::DataOutputStream</b> (A data output stream lets an application write primitive Java data types to an output stream in a portable way ) . . . . .	1628
<b>activemq::commands::DataResponse</b> . . . . .	1632
<b>activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>DataResponseMarshaller</b> (p. 1635) ) . . . . .	1635
<b>activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>DataResponseMarshaller</b> (p. 1639) ) . . . . .	1639
<b>activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>DataResponseMarshaller</b> (p. 1643) ) . . . . .	1643

<b>activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>DataResponseMarshaller</b> (p. 1647) ) . . . . .	1647
<b>activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>DataResponseMarshaller</b> (p. 1652) ) . . . . .	1652
<b>activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>DataResponseMarshaller</b> (p. 1656) ) . . . . .	1656
<b>activemq::wireformat::openwire::marshal::DataStreamMarshaller</b> (Base class for all classes that marshal commands for Openwire ) . . . . .	1660
<b>activemq::commands::DataStructure</b> . . . . .	1713
<b>decaf::util::Date</b> (Wrapper class around a time value in milliseconds ) . . . . .	1719
<b>decaf::internal::DecafRuntime</b> (Handles APR initialization and termination ) . . . . .	1723
<b>activemq::threads::DedicatedTaskRunner</b> . . . . .	1724
<b>activemq::core::policies::DefaultPrefetchPolicy</b> . . . . .	1726
<b>activemq::core::policies::DefaultRedeliveryPolicy</b> . . . . .	1730
<b>decaf::internal::net::DefaultServerSocketFactory</b> (Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options ) . . . . .	1735
<b>decaf::internal::net::DefaultSocketFactory</b> (SocketFactory implementation that is used to create Sockets ) . . . . .	1738
<b>decaf::internal::net::ssl::DefaultSSLContext</b> (Default SSLContext manager for the Decaf library ) . . . . .	1743
<b>decaf::internal::net::ssl::DefaultSSLServerSocketFactory</b> (Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds ) . . . . .	1744
<b>decaf::internal::net::ssl::DefaultSSLSocketFactory</b> (Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds ) . . . . .	1749
<b>activemq::transport::DefaultTransportListener</b> . . . . .	1757
<b>decaf::util::zip::Deflater</b> (This class compresses data using the <i>DEFLATE</i> algorithm (see <i>specification</i> ) ) . . . . .	1759
<b>decaf::util::zip::DeflaterOutputStream</b> (Provides a FilterOutputStream instance that compresses the data before writing it to the wrapped OutputStream ) . . . . .	1769
<b>decaf::util::concurrent::Delayed</b> (A mix-in style interface for marking objects that should be acted upon after a given delay ) . . . . .	1774
<b>cms::DeliveryMode</b> (This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages ) . . . . .	1775
<b>cms::Destination</b> (A <b>Destination</b> (p. 1776) object encapsulates a provider-specific address ) . . . . .	1776
<b>activemq::commands::ActiveMQDestination::DestinationFilter</b> . . . . .	1779
<b>activemq::commands::DestinationInfo</b> . . . . .	1779



<b>activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>DestinationInfoMarshaller</b> (p. 1784) ) . . . . .	1784
<b>activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>DestinationInfoMarshaller</b> (p. 1789) ) . . . . .	1789
<b>activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>DestinationInfoMarshaller</b> (p. 1793) ) . . . . .	1793
<b>activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>DestinationInfoMarshaller</b> (p. 1797) ) . . . . .	1797
<b>activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>DestinationInfoMarshaller</b> (p. 1801) ) . . . . .	1801
<b>activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>DestinationInfoMarshaller</b> (p. 1806) ) . . . . .	1806
<b>activemq::cmsutil::DestinationResolver</b> (Resolves a CMS destination name to a <i>Destination</i> ) . . . . .	1810
<b>activemq::commands::DiscoveryEvent</b> . . . . .	1812
<b>activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller</b> (Marshaling code for Open Wire Format for <b>DiscoveryEventMarshaller</b> (p. 1815) ) . . . . .	1815
<b>activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller</b> (Marshaling code for Open Wire Format for <b>DiscoveryEventMarshaller</b> (p. 1819) ) . . . . .	1819
<b>activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller</b> (Marshaling code for Open Wire Format for <b>DiscoveryEventMarshaller</b> (p. 1823) ) . . . . .	1823
<b>activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller</b> (Marshaling code for Open Wire Format for <b>DiscoveryEventMarshaller</b> (p. 1827) ) . . . . .	1827
<b>activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller</b> (Marshaling code for Open Wire Format for <b>DiscoveryEventMarshaller</b> (p. 1831) ) . . . . .	1831
<b>activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller</b> (Marshaling code for Open Wire Format for <b>DiscoveryEventMarshaller</b> (p. 1835) ) . . . . .	1835
<b>activemq::core::DispatchData</b> (Simple POJO that contains the information necessary to route a message to a specified consumer) . . . . .	1839
<b>activemq::core::Dispatcher</b> (Interface for an object responsible for dispatching messages to consumers) . . . . .	1840
<b>decaf::lang::Double</b> . . . . .	1841
<b>decaf::internal::nio::DoubleArrayBuffer</b> . . . . .	1853
<b>decaf::nio::DoubleBuffer</b> (This class defines four categories of operations upon double buffers: ) . . . . .	1865
<b>decaf::lang::DYNAMIC_CAST_TOKEN</b> . . . . .	1878
<b>activemq::cmsutil::DynamicDestinationResolver</b> (Resolves a CMS destination name to a <i>Destination</i> ) . . . . .	1878

<b>decaf::util::Map&lt; K, V, COMPARATOR &gt;::Entry</b>	1880
<b>decaf::io::EOFException</b>	1881
<b>decaf::util::logging::ErrorManager</b> ( <b>ErrorManager</b> (p. 1884) objects can be attached to Handlers to process any error that occur on a <b>Handler</b> (p. 2042) during Logging )	1884
<b>decaf::lang::Exception</b>	1886
<b>cms::ExceptionListener</b> (If a CMS provider detects a serious problem, it notifies the client application through an <b>ExceptionListener</b> (p. 1893) that is registered with the <b>Connection</b> (p. 1296) )	1893
<b>activemq::commands::ExceptionResponse</b>	1894
<b>activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ExceptionResponseMarshaller</b> (p. 1898) )	1898
<b>activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ExceptionResponseMarshaller</b> (p. 1902) )	1902
<b>activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ExceptionResponseMarshaller</b> (p. 1906) )	1906
<b>activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ExceptionResponseMarshaller</b> (p. 1910) )	1910
<b>activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ExceptionResponseMarshaller</b> (p. 1915) )	1915
<b>activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ExceptionResponseMarshaller</b> (p. 1919) )	1919
<b>decaf::util::concurrent::ExecutionException</b>	1923
<b>decaf::util::concurrent::Executor</b> (An object that executes submitted <b>decaf.lang Runnable</b> (p. 3418) tasks )	1926
<b>decaf::util::concurrent::ExecutorService</b> (An <b>Executor</b> (p. 1926) that provides methods to manage termination and methods that can produce a <b>Future</b> (p. 2029) for tracking progress of one or more asynchronous tasks )	1928
<b>activemq::transport::failover::FailoverTransport</b>	1930
<b>activemq::transport::failover::FailoverTransportFactory</b> (Creates an instance of a <b>FailoverTransport</b> (p. 1930) )	1942
<b>activemq::transport::failover::FailoverTransportListener</b> (Utility class used by the <b>Transport</b> (p. 3996) to perform the work of responding to events from the active <b>Transport</b> (p. 3996) )	1945
<b>decaf::io::FileDescriptor</b> (This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files )	1947
<b>decaf::util::logging::Filter</b> (A <b>Filter</b> (p. 1949) can be used to provide fine grain control over what is logged, beyond the control provided by log levels )	1949

<b>decaf::io::FilterInputStream</b> (A <b>FilterInputStream</b> (p. 1950) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality ) . . . . .	1950
<b>decaf::io::FilterOutputStream</b> (This class is the superclass of all classes that filter output streams ) . . . . .	1957
<b>decaf::lang::Float</b> . . . . .	1961
<b>decaf::internal::nio::FloatArrayBuffer</b> . . . . .	1974
<b>decaf::nio::FloatBuffer</b> (This class defines four categories of operations upon float buffers: ) . . . . .	1985
<b>decaf::io::Flushable</b> (A <b>Flushable</b> (p. 1998) is a destination of data that can be flushed ) . . . . .	1998
<b>activemq::commands::FlushCommand</b> . . . . .	1999
<b>activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>FlushCommandMarshaller</b> (p. 2002) ) . . . . .	2002
<b>activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>FlushCommandMarshaller</b> (p. 2006) ) . . . . .	2006
<b>activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>FlushCommandMarshaller</b> (p. 2010) ) . . . . .	2010
<b>activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>FlushCommandMarshaller</b> (p. 2014) ) . . . . .	2014
<b>activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>FlushCommandMarshaller</b> (p. 2019) ) . . . . .	2019
<b>activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>FlushCommandMarshaller</b> (p. 2023) ) . . . . .	2023
<b>decaf::util::logging::Formatter</b> (A <b>Formatter</b> (p. 2027) provides support for formatting LogRecords ) . . . . .	2027
<b>decaf::util::concurrent::Future&lt; V &gt;</b> (A <b>Future</b> (p. 2029) represents the result of an asynchronous computation ) . . . . .	2029
<b>activemq::transport::correlator::FutureResponse</b> (A container that holds a response object ) . . . . .	2032
<b>decaf::security::GeneralSecurityException</b> . . . . .	2034
<b>decaf::internal::util::GenericResource&lt; T &gt;</b> (A <b>Generic Resource</b> (p. 3374) wraps some type and will delete it when the <b>Resource</b> (p. 3374) itself is deleted ) . . . . .	2037
<b>gz_header_s</b> . . . . .	2038
<b>gz_state</b> . . . . .	2039
<b>decaf::util::logging::Handler</b> (A <b>Handler</b> (p. 2042) object takes log messages from a <b>Logger</b> (p. 2461) and exports them ) . . . . .	2042
<b>decaf::internal::util::HexStringParser</b> . . . . .	2046
<b>activemq::wireformat::openwire::utils::HexTable</b> (Maps hexadecimal strings to the value of an index into the table, i.e ) . . . . .	2048
<b>decaf::net::HttpRetryException</b> . . . . .	2049
<b>activemq::util::IdGenerator</b> . . . . .	2052

<b>decaf::lang::exceptions::IllegalArgumentException</b>	2054
<b>decaf::lang::exceptions::IllegalMonitorStateException</b>	2056
<b>cms::IllegalStateException</b> (This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation )	2059
<b>decaf::lang::exceptions::IllegalStateException</b>	2060
<b>decaf::lang::exceptions::IllegalThreadStateException</b>	2063
<b>activemq::transport::inactivity::InactivityMonitor</b>	2065
<b>decaf::lang::exceptions::IndexOutOfBoundsException</b>	2069
<b>decaf::net::Inet4Address</b>	2072
<b>decaf::net::Inet6Address</b>	2076
<b>decaf::net::InetAddress</b> (Represents an IP address )	2077
<b>decaf::net::InetSocketAddress</b>	2085
<b>inflate_state</b>	2085
<b>decaf::util::zip::Inflater</b> (This class uncompresses data that was compressed using the <i>DEFLATE</i> algorithm (see <i>specification</i> ))	2088
<b>decaf::util::zip::InflaterInputStream</b> (A <i>FilterInputStream</i> that decompresses data read from the wrapped <i>InputStream</i> instance )	2097
<b>decaf::io::InputStream</b> (A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes )	2105
<b>decaf::io::InputStreamReader</b> (An <i>InputStreamReader</i> (p. 2116) is a bridge from byte streams to character streams )	2116
<b>decaf::internal::nio::IntArrayBuffer</b>	2119
<b>decaf::nio::IntBuffer</b> (This class defines four categories of operations upon <i>int</i> buffers: )	2130
<b>decaf::lang::Integer</b>	2143
<b>activemq::commands::IntegerResponse</b>	2160
<b>activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller</b> (Marshaling code for Open Wire Format for <i>IntegerResponseMarshaller</i> (p. 2162) )	2162
<b>activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller</b> (Marshaling code for Open Wire Format for <i>IntegerResponseMarshaller</i> (p. 2167) )	2167
<b>activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller</b> (Marshaling code for Open Wire Format for <i>IntegerResponseMarshaller</i> (p. 2171) )	2171
<b>activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller</b> (Marshaling code for Open Wire Format for <i>IntegerResponseMarshaller</i> (p. 2175) )	2175
<b>activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller</b> (Marshaling code for Open Wire Format for <i>IntegerResponseMarshaller</i> (p. 2180) )	2180
<b>activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller</b> (Marshaling code for Open Wire Format for <i>IntegerResponseMarshaller</i> (p. 2184) )	2184
<b>internal_state</b>	2188
<b>activemq::transport::mock::InternalCommandListener</b> (Listens for Commands sent from the <i>MockTransport</i> (p. 2854) )	2192
<b>decaf::lang::exceptions::InterruptedException</b>	2193
<b>decaf::io::InterruptedException</b>	2196

<b>cms::InvalidClientIdException</b> (This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider ) . . . . .	2199
<b>cms::InvalidDestinationException</b> (This exception must be thrown when a destination either is not understood by a provider or is no longer valid ) . . . . .	2200
<b>decaf::security::InvalidKeyException</b> . . . . .	2201
<b>decaf::nio::InvalidMarkException</b> . . . . .	2204
<b>cms::InvalidSelectorException</b> (This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax ) . . . . .	2206
<b>decaf::lang::exceptions::InvalidStateException</b> . . . . .	2207
<b>decaf::io::IOException</b> . . . . .	2210
<b>activemq::transport::IOTransport</b> (Implementation of the <b>Transport</b> (p. 3996) interface that performs marshaling of commands to IO streams ) . . . . .	2213
<b>decaf::lang::Iterable&lt; E &gt;</b> (Implementing this interface allows an object to be cast to an <b>Iterable</b> (p. 2220) type for generic collections API calls ) . . . . .	2220
<b>decaf::util::Iterator&lt; T &gt;</b> (Defines an object that can be used to iterate over the elements of a collection ) . . . . .	2222
<b>activemq::commands::JournalQueueAck</b> . . . . .	2224
<b>activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalQueueAckMarshaller</b> (p. 2228) ) . . . . .	2228
<b>activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalQueueAckMarshaller</b> (p. 2232) ) . . . . .	2232
<b>activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalQueueAckMarshaller</b> (p. 2236) ) . . . . .	2236
<b>activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalQueueAckMarshaller</b> (p. 2240) ) . . . . .	2240
<b>activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalQueueAckMarshaller</b> (p. 2244) ) . . . . .	2244
<b>activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalQueueAckMarshaller</b> (p. 2248) ) . . . . .	2248
<b>activemq::commands::JournalTopicAck</b> . . . . .	2252
<b>activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTopicAckMarshaller</b> (p. 2256) ) . . . . .	2256
<b>activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTopicAckMarshaller</b> (p. 2261) ) . . . . .	2261
<b>activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTopicAckMarshaller</b> (p. 2265) ) . . . . .	2265

<b>activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTopicAck-</b> <b>Marshaller</b> (p. 2269) ) . . . . .	2269
<b>activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTopicAck-</b> <b>Marshaller</b> (p. 2273) ) . . . . .	2273
<b>activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTopicAck-</b> <b>Marshaller</b> (p. 2277) ) . . . . .	2277
<b>activemq::commands::JournalTrace</b> . . . . .	2281
<b>activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTraceMar-</b> <b>shaller</b> (p. 2283) ) . . . . .	2283
<b>activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTraceMar-</b> <b>shaller</b> (p. 2288) ) . . . . .	2288
<b>activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTraceMar-</b> <b>shaller</b> (p. 2292) ) . . . . .	2292
<b>activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTraceMar-</b> <b>shaller</b> (p. 2296) ) . . . . .	2296
<b>activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTraceMar-</b> <b>shaller</b> (p. 2300) ) . . . . .	2300
<b>activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTraceMar-</b> <b>shaller</b> (p. 2304) ) . . . . .	2304
<b>activemq::commands::JournalTransaction</b> . . . . .	2308
<b>activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTransaction-</b> <b>Marshaller</b> (p. 2311) ) . . . . .	2311
<b>activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTransaction-</b> <b>Marshaller</b> (p. 2315) ) . . . . .	2315
<b>activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTransaction-</b> <b>Marshaller</b> (p. 2319) ) . . . . .	2319
<b>activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTransaction-</b> <b>Marshaller</b> (p. 2323) ) . . . . .	2323
<b>activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTransaction-</b> <b>Marshaller</b> (p. 2327) ) . . . . .	2327
<b>activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTransaction-</b> <b>Marshaller</b> (p. 2331) ) . . . . .	2331
<b>activemq::commands::KeepAliveInfo</b> . . . . .	2335

<b>activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>KeepAliveInfoMarshaller</b> (p. 2338) ) . . . . .	2338
<b>activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>KeepAliveInfoMarshaller</b> (p. 2342) ) . . . . .	2342
<b>activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>KeepAliveInfoMarshaller</b> (p. 2347) ) . . . . .	2347
<b>activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>KeepAliveInfoMarshaller</b> (p. 2351) ) . . . . .	2351
<b>activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>KeepAliveInfoMarshaller</b> (p. 2355) ) . . . . .	2355
<b>activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>KeepAliveInfoMarshaller</b> (p. 2359) ) . . . . .	2359
<b>decaf::security::Key</b> (The <b>Key</b> (p. 2364) interface is the top-level interface for all keys ) . . . . .	2364
<b>decaf::security::KeyException</b> . . . . .	2366
<b>decaf::security::KeyManagementException</b> . . . . .	2368
<b>activemq::commands::LastPartialCommand</b> . . . . .	2371
<b>activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>LastPartialCommandMarshaller</b> (p. 2374) ) . . . . .	2374
<b>activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>LastPartialCommandMarshaller</b> (p. 2378) ) . . . . .	2378
<b>activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>LastPartialCommandMarshaller</b> (p. 2382) ) . . . . .	2382
<b>activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>LastPartialCommandMarshaller</b> (p. 2386) ) . . . . .	2386
<b>activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>LastPartialCommandMarshaller</b> (p. 2391) ) . . . . .	2391
<b>activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>LastPartialCommandMarshaller</b> (p. 2395) ) . . . . .	2395
<b>decaf::util::comparators::Less&lt; E &gt;</b> (Simple <b>Less</b> (p. 2399) <b>Comparator</b> (p. 1251) that compares to elements to determine if the first is less than the second ) . . . . .	2399
<b>std::less&lt; decaf::lang::ArrayPointer&lt; T &gt; &gt;</b> (An override of the less function object so that the Pointer objects can be stored in STL Maps, etc ) . . . . .	2401
<b>std::less&lt; decaf::lang::Pointer&lt; T &gt; &gt;</b> (An override of the less function object so that the Pointer objects can be stored in STL Maps, etc ) . . . . .	2402

<b>decaf::util::logging::Level</b> (Defines a set of standard logging levels that can be used to control logging output ) . . . . .	2403
<b>decaf::util::List&lt; E &gt;</b> (An ordered collection (also known as a sequence) ) .	2409
<b>decaf::util::ListIterator&lt; E &gt;</b> (An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list ) . . . .	2417
<b>activemq::commands::LocalTransactionId</b> . . . . .	2420
<b>activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>LocalTransactionIdMarshaller</b> (p. 2425) ) . . . . .	2425
<b>activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>LocalTransactionIdMarshaller</b> (p. 2429) ) . . . . .	2429
<b>activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>LocalTransactionIdMarshaller</b> (p. 2433) ) . . . . .	2433
<b>activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>LocalTransactionIdMarshaller</b> (p. 2437) ) . . . . .	2437
<b>activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>LocalTransactionIdMarshaller</b> (p. 2442) ) . . . . .	2442
<b>activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>LocalTransactionIdMarshaller</b> (p. 2446) ) . . . . .	2446
<b>decaf::util::concurrent::Lock</b> (A wrapper class around a given synchronization mechanism that provides automatic release upon destruction ) . . . . .	2450
<b>decaf::util::concurrent::locks::Lock</b> ( <b>Lock</b> (p. 2452) implementations provide more extensive locking operations than can be obtained using synchronized statements ) . . . . .	2452
<b>decaf::util::concurrent::locks::LockSupport</b> (Basic thread blocking primitives for creating locks and other synchronization classes ) . . . .	2458
<b>decaf::util::logging::Logger</b> (A <b>Logger</b> (p. 2461) object is used to log messages for a specific system or application component ) . . . . .	2461
<b>decaf::util::logging::LoggerHierarchy</b> . . . . .	2474
<b>activemq::io::LoggingInputStream</b> . . . . .	2474
<b>activemq::io::LoggingOutputStream</b> (OutputStream filter that just logs the data being written ) . . . . .	2476
<b>activemq::transport::logging::LoggingTransport</b> (A transport filter that logs commands as they are sent/received ) . . . . .	2477
<b>decaf::util::logging::LogManager</b> (There is a single global <b>LogManager</b> (p. 2480) object that is used to maintain a set of shared state about Loggers and log services ) . . . . .	2480
<b>decaf::util::logging::LogRecord</b> ( <b>LogRecord</b> (p. 2487) objects are used to pass logging requests between the logging framework and individual log Handlers ) . . . . .	2487
<b>decaf::util::logging::LogWriter</b> . . . . .	2493
<b>decaf::lang::Long</b> . . . . .	2495
<b>decaf::internal::nio::LongArrayBuffer</b> . . . . .	2511



<b>decaf::nio::LongBuffer</b> (This class defines four categories of operations upon long long buffers: ) . . . . .	2522
<b>activemq::util::LongSequenceGenerator</b> (This class is used to generate a sequence of long long values that are incremented each time a new value is requested ) . . . . .	2535
<b>decaf::net::MalformedURLException</b> . . . . .	2536
<b>decaf::util::Map&lt; K, V, COMPARATOR &gt;</b> ( <b>Map</b> (p. 2538) template that wraps around a <code>std::map</code> to provide a more user-friendly interface and to provide common functions that do not exist in <code>std::map</code> ) . . .	2538
<b>cms::MapMessage</b> (A <b>MapMessage</b> (p. 2551) object is used to send a set of name-value pairs ) . . . . .	2551
<b>decaf::util::logging::MarkBlockLogger</b> (Defines a class that can be used to mark the entry and exit from scoped blocks ) . . . . .	2563
<b>activemq::wireformat::MarshalAware</b> . . . . .	2564
<b>activemq::wireformat::openwire::marshal::v6::MarshallerFactory</b> (Used to create marshallers for a specific version of the wire protocol ) . .	2567
<b>activemq::wireformat::openwire::marshal::v3::MarshallerFactory</b> (Used to create marshallers for a specific version of the wire protocol ) . .	2567
<b>activemq::wireformat::openwire::marshal::v4::MarshallerFactory</b> (Used to create marshallers for a specific version of the wire protocol ) . .	2568
<b>activemq::wireformat::openwire::marshal::v5::MarshallerFactory</b> (Used to create marshallers for a specific version of the wire protocol ) . .	2569
<b>activemq::wireformat::openwire::marshal::v1::MarshallerFactory</b> (Used to create marshallers for a specific version of the wire protocol ) . .	2570
<b>activemq::wireformat::openwire::marshal::v2::MarshallerFactory</b> (Used to create marshallers for a specific version of the wire protocol ) . .	2570
<b>activemq::util::MarshallingSupport</b> . . . . .	2571
<b>decaf::lang::Math</b> (The class <b>Math</b> (p. 2575) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions ) . . . . .	2575
<b>activemq::util::MemoryUsage</b> . . . . .	2592
<b>activemq::commands::Message</b> . . . . .	2596
<b>cms::Message</b> (Root of all messages ) . . . . .	2614
<b>activemq::commands::MessageAck</b> . . . . .	2643
<b>activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageAckMarshaller</b> (p. 2648) ) . . . . .	2648
<b>activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageAckMarshaller</b> (p. 2653) ) . . . . .	2653
<b>activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageAckMarshaller</b> (p. 2657) ) . . . . .	2657
<b>activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageAckMarshaller</b> (p. 2661) ) . . . . .	2661
<b>activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageAckMarshaller</b> (p. 2665) ) . . . . .	2665

<b>activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageAckMarshaller</b> (p. 2670) ) . . . . .	2670
<b>cms::MessageConsumer</b> (A client uses a <b>MessageConsumer</b> (p. 2674) to received messages from a destination ) . . . . .	2674
<b>activemq::cmsutil::MessageCreator</b> (Creates the user-defined message to be sent by the <b>CmsTemplate</b> (p. 1201) ) . . . . .	2677
<b>activemq::commands::MessageDispatch</b> . . . . .	2678
<b>activemq::core::MessageDispatchChannel</b> . . . . .	2683
<b>activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchMarshaller</b> (p. 2690) ) . . . . .	2690
<b>activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchMarshaller</b> (p. 2695) ) . . . . .	2695
<b>activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchMarshaller</b> (p. 2699) ) . . . . .	2699
<b>activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchMarshaller</b> (p. 2703) ) . . . . .	2703
<b>activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchMarshaller</b> (p. 2707) ) . . . . .	2707
<b>activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchMarshaller</b> (p. 2712) ) . . . . .	2712
<b>activemq::commands::MessageDispatchNotification</b> . . . . .	2716
<b>activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchNotificationMarshaller</b> (p. 2720) ) . . . . .	2720
<b>activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchNotificationMarshaller</b> (p. 2725) ) . . . . .	2725
<b>activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchNotificationMarshaller</b> (p. 2729) ) . . . . .	2729
<b>activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchNotificationMarshaller</b> (p. 2733) ) . . . . .	2733
<b>activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchNotificationMarshaller</b> (p. 2737) ) . . . . .	2737
<b>activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchNotificationMarshaller</b> (p. 2742) ) . . . . .	2742
<b>cms::MessageEnumeration</b> (Defines an object that enumerates a collection of Messages ) . . . . .	2746

<b>cms::MessageEOFException</b> (This exception must be thrown when an unexpected end of stream has been reached when a <b>StreamMessage</b> (p. 3760) or <b>BytesMessage</b> (p. 1079) is being read ) . . . . .	2747
<b>cms::MessageFormatException</b> (This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type ) . . . . .	2749
<b>activemq::commands::MessageId</b> . . . . .	2750
<b>activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageIdMarshaller</b> (p. 2755) ) . . . . .	2755
<b>activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageIdMarshaller</b> (p. 2759) ) . . . . .	2759
<b>activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageIdMarshaller</b> (p. 2763) ) . . . . .	2763
<b>activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageIdMarshaller</b> (p. 2767) ) . . . . .	2767
<b>activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageIdMarshaller</b> (p. 2771) ) . . . . .	2771
<b>activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageIdMarshaller</b> (p. 2775) ) . . . . .	2775
<b>cms::MessageListener</b> (A <b>MessageListener</b> (p. 2779) object is used to receive asynchronously delivered messages ) . . . . .	2779
<b>activemq::wireformat::openwire::marshal::v5::MessageMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageMarshaller</b> (p. 2780) ) . . . . .	2780
<b>activemq::wireformat::openwire::marshal::v3::MessageMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageMarshaller</b> (p. 2785) ) . . . . .	2785
<b>activemq::wireformat::openwire::marshal::v2::MessageMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageMarshaller</b> (p. 2789) ) . . . . .	2789
<b>activemq::wireformat::openwire::marshal::v4::MessageMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageMarshaller</b> (p. 2793) ) . . . . .	2793
<b>activemq::wireformat::openwire::marshal::v1::MessageMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageMarshaller</b> (p. 2798) ) . . . . .	2798
<b>activemq::wireformat::openwire::marshal::v6::MessageMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageMarshaller</b> (p. 2802) ) . . . . .	2802
<b>cms::MessageNotReadableException</b> (This exception must be thrown when a CMS client attempts to read a write-only message ) . . . . .	2807
<b>cms::MessageNotWritableException</b> (This exception must be thrown when a CMS client attempts to write to a read-only message ) . . . . .	2808

<b>cms::MessageProducer</b> (A client uses a <b>MessageProducer</b> (p. 2809) object to send messages to a <b>Destination</b> (p. 1776) ) . . . . .	2809
<b>activemq::wireformat::openwire::utils::MessagePropertyInterceptor</b> (Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties ) . . . . .	2817
<b>activemq::commands::MessagePull</b> . . . . .	2824
<b>activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller</b> (Marshaling code for Open Wire Format for <b>MessagePullMarshaller</b> (p. 2828) ) . . . . .	2828
<b>activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller</b> (Marshaling code for Open Wire Format for <b>MessagePullMarshaller</b> (p. 2833) ) . . . . .	2833
<b>activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller</b> (Marshaling code for Open Wire Format for <b>MessagePullMarshaller</b> (p. 2837) ) . . . . .	2837
<b>activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller</b> (Marshaling code for Open Wire Format for <b>MessagePullMarshaller</b> (p. 2841) ) . . . . .	2841
<b>activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller</b> (Marshaling code for Open Wire Format for <b>MessagePullMarshaller</b> (p. 2845) ) . . . . .	2845
<b>activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller</b> (Marshaling code for Open Wire Format for <b>MessagePullMarshaller</b> (p. 2850) ) . . . . .	2850
<b>activemq::transport::mock::MockTransport</b> (The <b>MockTransport</b> (p. 2854) defines a base level <b>Transport</b> (p. 3996) class that is intended to be used in place of an a regular protocol <b>Transport</b> (p. 3996) such as TCP ) . . . . .	2854
<b>activemq::transport::mock::MockTransportFactory</b> (Manufactures Mock-Transports, which are objects that read from input streams and write to output streams ) . . . . .	2864
<b>decaf::util::concurrent::Mutex</b> ( <b>Mutex</b> (p. 2866) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java ) . . . .	2866
<b>decaf::util::concurrent::MutexHandle</b> . . . . .	2871
<b>decaf::internal::util::concurrent::MutexImpl</b> . . . . .	2872
<b>decaf::internal::net::Network</b> (Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API ) . . . . .	2874
<b>activemq::commands::NetworkBridgeFilter</b> . . . . .	2877
<b>activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller</b> (Marshaling code for Open Wire Format for <b>NetworkBridgeFilterMarshaller</b> (p. 2881) ) . . . . .	2881
<b>activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller</b> (Marshaling code for Open Wire Format for <b>NetworkBridgeFilterMarshaller</b> (p. 2885) ) . . . . .	2885

<b>activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller</b> (Marshaling code for Open Wire Format for <b>NetworkBridgeFilter-</b> <b>Marshaller</b> (p. 2889) ) . . . . .	2889
<b>activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller</b> (Marshaling code for Open Wire Format for <b>NetworkBridgeFilter-</b> <b>Marshaller</b> (p. 2893) ) . . . . .	2893
<b>activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller</b> (Marshaling code for Open Wire Format for <b>NetworkBridgeFilter-</b> <b>Marshaller</b> (p. 2897) ) . . . . .	2897
<b>activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller</b> (Marshaling code for Open Wire Format for <b>NetworkBridgeFilter-</b> <b>Marshaller</b> (p. 2901) ) . . . . .	2901
<b>decaf::net::NoRouteToHostException</b> . . . . .	2905
<b>decaf::security::NoSuchAlgorithmException</b> . . . . .	2907
<b>decaf::lang::exceptions::NoSuchElementException</b> . . . . .	2910
<b>decaf::security::NoSuchProviderException</b> . . . . .	2913
<b>decaf::lang::exceptions::NullPointerException</b> . . . . .	2915
<b>decaf::lang::Number</b> (The abstract class <b>Number</b> (p. 2918) is the super- class of classes <b>Byte</b> (p. 969), <b>Double</b> (p. 1841), <b>Float</b> (p. 1961), <b>Integer</b> (p. 2143), <b>Long</b> (p. 2495), and <b>Short</b> (p. 3538) ) . . . . .	2918
<b>decaf::lang::exceptions::NumberFormatException</b> . . . . .	2921
<b>cms::ObjectMessage</b> (Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object ) . . . . .	2924
<b>decaf::internal::net::ssl::openssl::OpenSSLContextSpi</b> (Provides an SSLCon- text that wraps the OpenSSL API ) . . . . .	2924
<b>decaf::internal::net::ssl::openssl::OpenSSLParameters</b> (Container class for parameters that are Common to OpenSSL socket classes ) . . . .	2927
<b>decaf::internal::net::ssl::openssl::OpenSSLServerSocket</b> (SSLServerSocket based on OpenSSL library code ) . . . . .	2929
<b>decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory</b> (SSLServer- SocketFactory that creates Server Sockets that use OpenSSL ) . . .	2935
<b>decaf::internal::net::ssl::openssl::OpenSSLSocket</b> (Wraps a a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API ) . . . . .	2940
<b>decaf::internal::net::ssl::openssl::OpenSSLSocketException</b> (Subclass of the standard SocketException that knows how to produce an error message from the OpenSSL error stack ) . . . . .	2955
<b>decaf::internal::net::ssl::openssl::OpenSSLSocketFactory</b> (Client Socket Factory that creates SSL based client sockets using the OpenSSL li- brary ) . . . . .	2959
<b>decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream</b> (An out- put stream for reading data from an OpenSSL Socket instance ) . . .	2966
<b>decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream</b> (Output- Stream implementation used to write data to an <b>OpenSSLSocket</b> (p. 2940) instance ) . . . . .	2969
<b>activemq::wireformat::openwire::OpenWireFormat</b> . . . . .	2971
<b>activemq::wireformat::openwire::OpenWireFormatFactory</b> . . . . .	2984
<b>activemq::wireformat::openwire::OpenWireFormatNegotiator</b> . . . . .	2985
<b>activemq::wireformat::openwire::OpenWireResponseBuilder</b> . . . . .	2989

<b>decaf::io::OutputStream</b> (Base interface for any class that wants to represent an output stream of bytes ) . . . . .	2991
<b>decaf::io::OutputStreamWriter</b> (A class for turning a character stream into a byte stream ) . . . . .	2999
<b>activemq::commands::PartialCommand</b> . . . . .	3002
<b>activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>PartialCommandMarshaller</b> (p. 3005) ) . . . . .	3005
<b>activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>PartialCommandMarshaller</b> (p. 3010) ) . . . . .	3010
<b>activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>PartialCommandMarshaller</b> (p. 3014) ) . . . . .	3014
<b>activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>PartialCommandMarshaller</b> (p. 3019) ) . . . . .	3019
<b>activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>PartialCommandMarshaller</b> (p. 3023) ) . . . . .	3023
<b>activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>PartialCommandMarshaller</b> (p. 3028) ) . . . . .	3028
<b>decaf::lang::Pointer&lt; T, REFCOUNTER &gt;</b> (Decaf's implementation of a Smart <b>Pointer</b> (p. 3032) that is a template on a Type and is <b>Thread</b> (p. 3876) Safe if the default Reference Counter is used ) . . . . .	3032
<b>decaf::lang::PointerComparator&lt; T, R &gt;</b> (This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the <b>Pointer</b> (p. 3032) instance ) . . . . .	3040
<b>activemq::cmsutil::PooledSession</b> (A pooled session object that wraps around a delegate session ) . . . . .	3041
<b>decaf::util::concurrent::PooledThread</b> . . . . .	3056
<b>decaf::util::concurrent::PooledThreadListener</b> (Abstract Listener Interface for users of <b>ThreadPool</b> (p. 3888) ) . . . . .	3058
<b>decaf::net::PortUnreachableException</b> . . . . .	3060
<b>activemq::core::PrefetchPolicy</b> (Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP ) . . . . .	3062
<b>activemq::util::PrimitiveList</b> (List of primitives ) . . . . .	3067
<b>activemq::util::PrimitiveMap</b> (Map of named primitives ) . . . . .	3079
<b>activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller</b> (This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire ) . . . . .	3090
<b>activemq::util::PrimitiveValueNode::PrimitiveValue</b> (Define a union type comprised of the various types ) . . . . .	3097

<b>activemq::util::PrimitiveValueConverter</b> (Class controls the conversion of data contained in a <b>PrimitiveValueNode</b> (p. 3099) from one type to another ) . . . . .	3098
<b>activemq::util::PrimitiveValueNode</b> (Class that wraps around a single value of one of the many types ) . . . . .	3099
<b>decaf::security::Principal</b> (Base interface for a principal, which can represent an individual or organization ) . . . . .	3114
<b>decaf::util::PriorityQueue&lt; E &gt;</b> (An unbounded priority queue based on a binary heap algorithm ) . . . . .	3116
<b>activemq::commands::ProducerAck</b> . . . . .	3124
<b>activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerAckMarshaller</b> (p. 3128) ) . . . . .	3128
<b>activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerAckMarshaller</b> (p. 3133) ) . . . . .	3133
<b>activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerAckMarshaller</b> (p. 3137) ) . . . . .	3137
<b>activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerAckMarshaller</b> (p. 3141) ) . . . . .	3141
<b>activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerAckMarshaller</b> (p. 3145) ) . . . . .	3145
<b>activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerAckMarshaller</b> (p. 3150) ) . . . . .	3150
<b>activemq::cmsutil::ProducerCallback</b> (Callback for sending a message to a CMS destination ) . . . . .	3154
<b>activemq::cmsutil::CmsTemplate::ProducerExecutor</b> . . . . .	3155
<b>activemq::commands::ProducerId</b> . . . . .	3157
<b>activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerIdMarshaller</b> (p. 3161) ) . . . . .	3161
<b>activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerIdMarshaller</b> (p. 3165) ) . . . . .	3165
<b>activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerIdMarshaller</b> (p. 3169) ) . . . . .	3169
<b>activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerIdMarshaller</b> (p. 3173) ) . . . . .	3173
<b>activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerIdMarshaller</b> (p. 3177) ) . . . . .	3177
<b>activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerIdMarshaller</b> (p. 3181) ) . . . . .	3181

<b>activemq::commands::ProducerInfo</b>	3185
<b>activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerInfoMarshaller</b> (p. 3190) )	3190
<b>activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerInfoMarshaller</b> (p. 3194) )	3194
<b>activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerInfoMarshaller</b> (p. 3199) )	3199
<b>activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerInfoMarshaller</b> (p. 3203) )	3203
<b>activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerInfoMarshaller</b> (p. 3207) )	3207
<b>activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerInfoMarshaller</b> (p. 3211) )	3211
<b>activemq::state::ProducerState</b>	3216
<b>decaf::util::Properties</b> (Java-like properties class for mapping string names to string values )	3216
<b>decaf::util::logging::PropertiesChangeListener</b> (Defines the interface that classes can use to listen for change events on <b>Properties</b> (p. 3216) )	3227
<b>decaf::net::ProtocolException</b>	3228
<b>decaf::security::PublicKey</b> (A public key )	3231
<b>decaf::io::PushbackInputStream</b> (A <b>PushbackInputStream</b> (p. 3231) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte )	3231
<b>cms::Queue</b> (An interface encapsulating a provider-specific queue name )	3238
<b>decaf::util::Queue&lt; E &gt;</b> (A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection )	3239
<b>cms::QueueBrowser</b> (This class implements in interface for browsing the messages in a <b>Queue</b> (p. 3238) without removing them )	3243
<b>decaf::util::Random</b> ( <b>Random</b> (p. 3245) Value Generator which is used to generate a stream of pseudorandom numbers )	3245
<b>decaf::lang::Readable</b> (A <b>Readable</b> (p. 3251) is a source of characters )	3251
<b>activemq::transport::inactivity::ReadChecker</b> (Runnable class that is used by the { )	3253
<b>decaf::io::Reader</b>	3254
<b>decaf::nio::ReadOnlyBufferException</b>	3260
<b>decaf::util::concurrent::locks::ReadWriteLock</b> (A <b>ReadWriteLock</b> (p. 3263) maintains a pair of associated locks, one for read-only operations and one for writing )	3263
<b>activemq::cmsutil::CmsTemplate::ReceiveExecutor</b>	3265
<b>activemq::core::RedeliveryPolicy</b> (Interface for a <b>RedeliveryPolicy</b> (p. 3267) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back )	3267



<b>decaf::util::concurrent::locks::ReentrantLock</b> (A reentrant mutual exclusion <b>Lock</b> (p. 2452) with extended capabilities ) . . . . .	3273
<b>decaf::util::concurrent::RejectedExecutionException</b> . . . . .	3280
<b>decaf::util::concurrent::RejectedExecutionHandler</b> (A handler for tasks that cannot be executed by a <b>ThreadPoolExecutor</b> (p. ??) ) . . . . .	3283
<b>activemq::commands::RemoveInfo</b> . . . . .	3284
<b>activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveInfoMarshaller</b> (p. 3288) ) . . . . .	3288
<b>activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveInfoMarshaller</b> (p. 3292) ) . . . . .	3292
<b>activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveInfoMarshaller</b> (p. 3296) ) . . . . .	3296
<b>activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveInfoMarshaller</b> (p. 3300) ) . . . . .	3300
<b>activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveInfoMarshaller</b> (p. 3305) ) . . . . .	3305
<b>activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveInfoMarshaller</b> (p. 3309) ) . . . . .	3309
<b>activemq::commands::RemoveSubscriptionInfo</b> . . . . .	3313
<b>activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveSubscriptionInfoMarshaller</b> (p. 3318) ) . . . . .	3318
<b>activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveSubscriptionInfoMarshaller</b> (p. 3322) ) . . . . .	3322
<b>activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveSubscriptionInfoMarshaller</b> (p. 3326) ) . . . . .	3326
<b>activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveSubscriptionInfoMarshaller</b> (p. 3331) ) . . . . .	3331
<b>activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveSubscriptionInfoMarshaller</b> (p. 3335) ) . . . . .	3335
<b>activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveSubscriptionInfoMarshaller</b> (p. 3339) ) . . . . .	3339
<b>activemq::commands::ReplayCommand</b> . . . . .	3343
<b>activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ReplayCommandMarshaller</b> (p. 3347) ) . . . . .	3347
<b>activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ReplayCommandMarshaller</b> (p. 3351) ) . . . . .	3351

<b>activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ReplayCommandMarshaller</b> (p. 3355) )	3355
<b>activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ReplayCommandMarshaller</b> (p. 3360) )	3360
<b>activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ReplayCommandMarshaller</b> (p. 3364) )	3364
<b>activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ReplayCommandMarshaller</b> (p. 3368) )	3368
<b>activemq::cmsutil::CmsTemplate::ResolveProducerExecutor</b>	3372
<b>activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor</b>	3373
<b>decaf::internal::util::Resource</b> (Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown )	3374
<b>decaf::internal::util::ResourceLifecycleManager</b>	3375
<b>activemq::cmsutil::ResourceLifecycleManager</b> (Manages the lifecycle of a set of CMS resources )	3376
<b>activemq::commands::Response</b>	3379
<b>activemq::transport::mock::ResponseBuilder</b> (Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol )	3382
<b>activemq::transport::correlator::ResponseCorrelator</b> (This type of transport filter is responsible for correlating asynchronous responses with requests )	3384
<b>activemq::wireformat::openwire::marshal::v4::ResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ResponseMarshaller</b> (p. 3388) )	3388
<b>activemq::wireformat::openwire::marshal::v2::ResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ResponseMarshaller</b> (p. 3393) )	3393
<b>activemq::wireformat::openwire::marshal::v5::ResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ResponseMarshaller</b> (p. 3398) )	3398
<b>activemq::wireformat::openwire::marshal::v3::ResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ResponseMarshaller</b> (p. 3403) )	3403
<b>activemq::wireformat::openwire::marshal::v1::ResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ResponseMarshaller</b> (p. 3408) )	3408
<b>activemq::wireformat::openwire::marshal::v6::ResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ResponseMarshaller</b> (p. 3413) )	3413
<b>decaf::lang::Runnable</b> (Interface for a runnable object - defines a task that can be run by a thread )	3418
<b>decaf::lang::Runtime</b>	3419
<b>decaf::lang::exceptions::RuntimeException</b>	3421
<b>decaf::security::SecureRandom</b>	3424

<b>decaf::internal::security::SecureRandomImpl</b> (Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources ) . . . . .	3429
<b>decaf::security::SecureRandomSpi</b> (Interface class used by Security Service Providers to implement a source of secure random bytes ) . . .	3432
<b>decaf::util::concurrent::Semaphore</b> (A counting semaphore ) . . . . .	3434
<b>activemq::cmsutil::CmsTemplate::SendExecutor</b> . . . . .	3445
<b>decaf::net::ServerSocket</b> (This class implements server sockets ) . . . . .	3447
<b>decaf::net::ServerSocketFactory</b> (Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies ) . . . . .	3457
<b>cms::Session</b> (A <b>Session</b> (p. 3460) object is a single-threaded context for producing and consuming messages ) . . . . .	3460
<b>activemq::cmsutil::SessionCallback</b> (Callback for executing any number of operations on a provided CMS Session ) . . . . .	3475
<b>activemq::commands::SessionId</b> . . . . .	3476
<b>activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionIdMarshaller</b> (p. 3481) ) . . . . .	3481
<b>activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionIdMarshaller</b> (p. 3485) ) . . . . .	3485
<b>activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionIdMarshaller</b> (p. 3489) ) . . . . .	3489
<b>activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionIdMarshaller</b> (p. 3493) ) . . . . .	3493
<b>activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionIdMarshaller</b> (p. 3497) ) . . . . .	3497
<b>activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionIdMarshaller</b> (p. 3501) ) . . . . .	3501
<b>activemq::commands::SessionInfo</b> . . . . .	3505
<b>activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionInfoMarshaller</b> (p. 3508) ) . . . . .	3508
<b>activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionInfoMarshaller</b> (p. 3513) ) . . . . .	3513
<b>activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionInfoMarshaller</b> (p. 3517) ) . . . . .	3517
<b>activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionInfoMarshaller</b> (p. 3521) ) . . . . .	3521
<b>activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionInfoMarshaller</b> (p. 3525) ) . . . . .	3525

<b>activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionInfoMarshaller</b> (p. 3530) ) . . . . .	3530
<b>activemq::cmsutil::SessionPool</b> (A pool of CMS sessions from the same connection and with the same acknowledge mode ) . . . . .	3534
<b>activemq::state::SessionState</b> . . . . .	3536
<b>decaf::util::Set&lt; E &gt;</b> (A collection that contains no duplicate elements ) . . . . .	3538
<b>decaf::lang::Short</b> . . . . .	3538
<b>decaf::internal::nio::ShortArrayBuffer</b> . . . . .	3548
<b>decaf::nio::ShortBuffer</b> (This class defines four categories of operations upon short buffers: ) . . . . .	3560
<b>activemq::commands::ShutdownInfo</b> . . . . .	3573
<b>activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ShutdownInfoMarshaller</b> (p. 3576) ) . . . . .	3576
<b>activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ShutdownInfoMarshaller</b> (p. 3580) ) . . . . .	3580
<b>activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ShutdownInfoMarshaller</b> (p. 3584) ) . . . . .	3584
<b>activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ShutdownInfoMarshaller</b> (p. 3589) ) . . . . .	3589
<b>activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ShutdownInfoMarshaller</b> (p. 3593) ) . . . . .	3593
<b>activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ShutdownInfoMarshaller</b> (p. 3597) ) . . . . .	3597
<b>decaf::security::SignatureException</b> . . . . .	3601
<b>decaf::util::logging::SimpleFormatter</b> (Print a brief summary of the <b>LogRecord</b> (p. 2487) in a human readable format ) . . . . .	3604
<b>decaf::util::logging::SimpleLogger</b> . . . . .	3605
<b>decaf::net::Socket</b> . . . . .	3607
<b>decaf::net::SocketAddress</b> (Base class for protocol specific <b>Socket</b> (p. 3607) addresses ) . . . . .	3626
<b>decaf::net::SocketError</b> (Static utility class to simplify handling of error codes for socket operations ) . . . . .	3626
<b>decaf::net::SocketException</b> (Exception for errors when manipulating sockets ) . . . . .	3627
<b>decaf::net::SocketFactory</b> (The <b>SocketFactory</b> (p. 3629) is used to create <b>Socket</b> (p. 3607) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations ) . . . . .	3629
<b>decaf::internal::net::SocketFileDescriptor</b> (File Descriptor type used internally by Decaf Socket objects ) . . . . .	3634
<b>decaf::net::SocketImpl</b> (Acts as a base class for all physical <b>Socket</b> (p. 3607) implementations ) . . . . .	3635
<b>decaf::net::SocketImplFactory</b> (Factory class interface for a Factory that creates <b>SocketImpl</b> objects ) . . . . .	3644

<b>decaf::net::SocketOptions</b>	3645
<b>decaf::net::SocketTimeoutException</b>	3650
<b>decaf::net::ssl::SSLContext</b> (Represents an implementation of the Secure Socket (p. 3607) Layer for streaming based sockets)	3653
<b>decaf::net::ssl::SSLContextSpi</b> (Defines the interface that should be provided by an <b>SSLContext</b> (p. 3653) provider)	3655
<b>decaf::net::ssl::SSLParameters</b>	3658
<b>decaf::net::ssl::SSLServerSocket</b> (Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol)	3662
<b>decaf::net::ssl::SSLServerSocketFactory</b> (Factory class interface that provides methods to create SSL Server Sockets)	3668
<b>decaf::net::ssl::SSLSocket</b>	3670
<b>decaf::net::ssl::SSLSocketFactory</b> (Factory class interface for a <b>SocketFactory</b> (p. 3629) that can create <b>SSLSocket</b> (p. 3670) objects)	3679
<b>activemq::transport::tcp::SslTransport</b> ( <b>Transport</b> (p. 3996) for connecting to a Broker using an SSL Socket)	3682
<b>activemq::transport::tcp::SslTransportFactory</b>	3684
<b>activemq::commands::BrokerError::StackTraceElement</b>	3685
<b>decaf::internal::io::StandardErrorOutputStream</b> (Wrapper Around the Standard error Output facility on the current platform)	3686
<b>decaf::internal::io::StandardInputStream</b>	3688
<b>decaf::internal::io::StandardOutputStream</b>	3689
<b>cms::Startable</b> (Interface for a class that implements the start method)	3691
<b>decaf::lang::STATIC_CAST_TOKEN</b>	3692
<b>activemq::core::ActiveMQConstants::StaticInitializer</b>	3693
<b>decaf::util::StlList&lt; E &gt;</b> ( <b>List</b> (p. 2409) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type)	3694
<b>decaf::util::StlMap&lt; K, V, COMPARATOR &gt;</b> ( <b>Map</b> (p. 2538) template that wraps around a <b>std::map</b> to provide a more user-friendly interface and to provide common functions that do not exist in <b>std::map</b> )	3708
<b>decaf::util::StlQueue&lt; T &gt;</b> (The <b>Queue</b> (p. 3239) class accepts messages with an <b>psuh(m)</b> command where <b>m</b> is the message to be queued)	3722
<b>decaf::util::StlSet&lt; E &gt;</b> ( <b>Set</b> (p. 3538) template that wraps around a <b>std::set</b> to provide a more user-friendly interface and to provide common functions that do not exist in <b>std::set</b> )	3731
<b>activemq::wireformat::stomp::StompCommandConstants</b>	3738
<b>activemq::wireformat::stomp::StompFrame</b> (A Stomp-level message frame that encloses all messages to and from the broker)	3741
<b>activemq::wireformat::stomp::StompHelper</b> (Utility Methods used when marshaling to and from <b>StompFrame</b> 's)	3746
<b>activemq::wireformat::stomp::StompWireFormat</b>	3751
<b>activemq::wireformat::stomp::StompWireFormatFactory</b> (Factory used to create the Stomp Wire Format instance)	3754
<b>cms::Stoppable</b> (Interface for a class that implements the stop method)	3755
<b>decaf::util::logging::StreamHandler</b> (Stream based logging <b>Handler</b> (p. 2042))	3756
<b>cms::StreamMessage</b> (Interface for a <b>StreamMessage</b> (p. 3760))	3760

<b>decaf::lang::String</b> (Immutable sequence of chars ) . . . . .	3775
<b>decaf::util::StringTokenizer</b> . . . . .	3779
<b>activemq::commands::SubscriptionInfo</b> . . . . .	3782
<b>activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SubscriptionInfoMarshaller</b> (p. 3786) ) . . . . .	3786
<b>activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SubscriptionInfoMarshaller</b> (p. 3791) ) . . . . .	3791
<b>activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SubscriptionInfoMarshaller</b> (p. 3795) ) . . . . .	3795
<b>activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SubscriptionInfoMarshaller</b> (p. 3799) ) . . . . .	3799
<b>activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SubscriptionInfoMarshaller</b> (p. 3803) ) . . . . .	3803
<b>activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SubscriptionInfoMarshaller</b> (p. 3807) ) . . . . .	3807
<b>decaf::util::concurrent::Synchronizable</b> (The interface for all synchronizable objects (that is, objects that can be locked and unlocked) ) . . .	3811
<b>decaf::internal::util::concurrent::SynchronizableImpl</b> (A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance) . . . .	3822
<b>activemq::core::Synchronization</b> (Transacted Object <b>Synchronization</b> (p. 3826), used to sync the events of a Transaction with the items in the Transaction) . . . . .	3826
<b>decaf::util::concurrent::SynchronousQueue&lt; E &gt;</b> (A <b>blocking queue</b> (p. 848) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa) . . . . .	3827
<b>decaf::lang::System</b> (Static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays) . . . . .	3838
<b>activemq::threads::Task</b> (Represents a unit of work that requires one or more iterations to complete) . . . . .	3846
<b>decaf::util::concurrent::TaskListener</b> . . . . .	3847
<b>activemq::threads::TaskRunner</b> . . . . .	3849
<b>decaf::internal::net::tcp::TcpSocket</b> (Platform-independent implementation of the socket interface) . . . . .	3850
<b>decaf::internal::net::tcp::TcpSocketInputStream</b> (Input stream for performing reads on a socket) . . . . .	3860
<b>decaf::internal::net::tcp::TcpSocketOutputStream</b> (Output stream for performing write operations on a socket) . . . . .	3863
<b>activemq::transport::tcp::TcpTransport</b> (Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an <b>IOTransport</b> (p. 2213) ) . . . . .	3865
<b>activemq::transport::tcp::TcpTransportFactory</b> (Factory Responsible for creating the <b>TcpTransport</b> (p. 3865) ) . . . . .	3869

<b>cms::TemporaryQueue</b> (Defines a Temporary <b>Queue</b> (p. 3238) based <b>Destination</b> (p. 1776) ) . . . . .	3871
<b>cms::TemporaryTopic</b> (Defines a Temporary <b>Topic</b> (p. 3930) based <b>Destination</b> (p. 1776) ) . . . . .	3873
<b>cms::TextMessage</b> (Interface for a text message ) . . . . .	3874
<b>decaf::lang::Thread</b> (A <b>Thread</b> (p. 3876) is a concurrent unit of execution ) . . . . .	3876
<b>decaf::util::concurrent::ThreadFactory</b> (Public interface <b>ThreadFactory</b> (p. 3887) ) . . . . .	3887
<b>decaf::lang::ThreadGroup</b> . . . . .	3888
<b>decaf::util::concurrent::ThreadPool</b> (Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks ) . . . . .	3888
<b>decaf::lang::Throwable</b> (This class represents an error that has occurred ) . . . . .	3895
<b>decaf::util::concurrent::TimeoutException</b> . . . . .	3899
<b>decaf::util::Timer</b> (A facility for threads to schedule tasks for future execution in a background thread ) . . . . .	3901
<b>decaf::util::TimerTask</b> (A Base class for a task object that can be scheduled for one-time or repeated execution by a <b>Timer</b> (p. 3901) ) . . . . .	3914
<b>decaf::internal::util::TimerTaskHeap</b> (A Binary Heap implemented specifically for the Timer class in Decaf Util ) . . . . .	3916
<b>decaf::util::concurrent::TimeUnit</b> (A <b>TimeUnit</b> (p. 3919) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units ) . . . . .	3919
<b>cms::Topic</b> (An interface encapsulating a provider-specific topic name ) . . . . .	3930
<b>activemq::state::Tracked</b> . . . . .	3931
<b>activemq::commands::TransactionId</b> . . . . .	3932
<b>activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionIdMarshaller</b> (p. 3935) ) . . . . .	3935
<b>activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionIdMarshaller</b> (p. 3939) ) . . . . .	3939
<b>activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionIdMarshaller</b> (p. 3943) ) . . . . .	3943
<b>activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionIdMarshaller</b> (p. 3947) ) . . . . .	3947
<b>activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionIdMarshaller</b> (p. 3951) ) . . . . .	3951
<b>activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionIdMarshaller</b> (p. 3955) ) . . . . .	3955
<b>activemq::commands::TransactionInfo</b> . . . . .	3959
<b>activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionInfoMarshaller</b> (p. 3964) ) . . . . .	3964

<b>activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionInfoMarshaller</b> (p. 3968) ) . . . . .	3968
<b>activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionInfoMarshaller</b> (p. 3972) ) . . . . .	3972
<b>activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionInfoMarshaller</b> (p. 3977) ) . . . . .	3977
<b>activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionInfoMarshaller</b> (p. 3981) ) . . . . .	3981
<b>activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionInfoMarshaller</b> (p. 3985) ) . . . . .	3985
<b>activemq::state::TransactionState</b> . . . . .	3989
<b>decaf::internal::util::concurrent::Transferer&lt; E &gt;</b> (Shared internal API for dual stacks and queues ) . . . . .	3991
<b>decaf::internal::util::concurrent::TransferQueue&lt; E &gt;</b> (This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers ) . . . . .	3992
<b>decaf::internal::util::concurrent::TransferStack&lt; E &gt;</b> . . . . .	3994
<b>activemq::transport::Transport</b> (Interface for a transport layer for command objects ) . . . . .	3996
<b>activemq::transport::TransportFactory</b> (Defines the interface for Factories that create Transports or TransportFilters ) . . . . .	4003
<b>activemq::transport::TransportFilter</b> (A filter on the transport layer ) . . . . .	4004
<b>activemq::transport::TransportListener</b> (A listener of asynchronous exceptions from a command transport object ) . . . . .	4013
<b>activemq::transport::TransportRegistry</b> (Registry of all <b>Transport</b> (p. 3996) Factories that are available to the client at runtime ) . . . . .	4015
<b>tree_desc_s</b> . . . . .	4018
<b>decaf::lang::Thread::UncaughtExceptionHandler</b> (Interface for handlers invoked when a <b>Thread</b> (p. 3876) abruptly terminates due to an uncaught exception ) . . . . .	4018
<b>decaf::net::UnknownHostException</b> . . . . .	4019
<b>decaf::net::UnknownServiceException</b> . . . . .	4022
<b>decaf::io::UnsupportedEncodingException</b> (Thrown when the the Character Encoding is not supported ) . . . . .	4025
<b>decaf::lang::exceptions::UnsupportedOperationException</b> . . . . .	4028
<b>cms::UnsupportedOperationException</b> (This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use ) . . . . .	4030
<b>decaf::net::URI</b> (This class represents an instance of a <b>URI</b> (p. 4031) as defined by RFC 2396 ) . . . . .	4031
<b>decaf::internal::net::URIEncoderDecoder</b> . . . . .	4044
<b>decaf::internal::net::URIHelper</b> (Helper class used by the URI classes in encoding and decoding of URI's ) . . . . .	4047
<b>activemq::transport::failover::URIPool</b> . . . . .	4054
<b>activemq::util::URISupport</b> . . . . .	4057



<b>decaf::net::URISyntaxException</b> . . . . .	4060
<b>decaf::internal::net::URIType</b> (Basic type object that holds data that com- poses a given URI ) . . . . .	4063
<b>decaf::net::URL</b> (Class <b>URL</b> (p. 4072) represents a Uniform Resource Lo- cator, a pointer to a "resource" on the World Wide Web ) . . . . .	4072
<b>decaf::net::URLDecoder</b> . . . . .	4074
<b>decaf::net::URLEncoder</b> . . . . .	4074
<b>activemq::util::Usage</b> . . . . .	4075
<b>decaf::io::UTFDataFormatException</b> (Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered ) . . . . .	4078
<b>decaf::util::UUID</b> (A class that represents an immutable universally unique identifier ( <b>UUID</b> (p. 4080)) ) . . . . .	4080
<b>activemq::wireformat::WireFormat</b> (Provides a mechanism to marshal com- mands into and out of packets or into and out of streams, Channels and Datagrams ) . . . . .	4088
<b>activemq::wireformat::WireFormatFactory</b> (The <b>WireFormatFactory</b> (p. 4092) is the interface that all <b>WireFormatFactory</b> (p. 4092) classes must extend ) . . . . .	4092
<b>activemq::commands::WireFormatInfo</b> . . . . .	4093
<b>activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>WireFormatInfoMar-</b> <b>shaller</b> (p. 4104) ) . . . . .	4104
<b>activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>WireFormatInfoMar-</b> <b>shaller</b> (p. 4109) ) . . . . .	4109
<b>activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>WireFormatInfoMar-</b> <b>shaller</b> (p. 4113) ) . . . . .	4113
<b>activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>WireFormatInfoMar-</b> <b>shaller</b> (p. 4117) ) . . . . .	4117
<b>activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>WireFormatInfoMar-</b> <b>shaller</b> (p. 4121) ) . . . . .	4121
<b>activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>WireFormatInfoMar-</b> <b>shaller</b> (p. 4125) ) . . . . .	4125
<b>activemq::wireformat::WireFormatNegotiator</b> (Defines a <b>WireFormat-</b> <b>Negotiator</b> (p. 4129) which allows a <b>WireFormat</b> (p. 4088) to ) . . . .	4129
<b>activemq::wireformat::WireFormatRegistry</b> (Registry of all <b>WireFormat</b> (p. 4088) Factories that are available to the client at runtime ) . . . .	4129
<b>activemq::transport::inactivity::WriteChecker</b> (Runnable class used by the { } ) . . . . .	4132
<b>decaf::io::Writer</b> . . . . .	4133
<b>decaf::security::auth::x500::X500Principal</b> . . . . .	4140
<b>decaf::security::cert::X509Certificate</b> (Base interface for all identity cer- tificates ) . . . . .	4141
<b>activemq::commands::XATransactionId</b> . . . . .	4143

<b>activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>XATransactionId-</b> <b>Marshaller</b> (p. 4147) ) . . . . .	4147
<b>activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>XATransactionId-</b> <b>Marshaller</b> (p. 4151) ) . . . . .	4151
<b>activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>XATransactionId-</b> <b>Marshaller</b> (p. 4155) ) . . . . .	4155
<b>activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>XATransactionId-</b> <b>Marshaller</b> (p. 4160) ) . . . . .	4160
<b>activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>XATransactionId-</b> <b>Marshaller</b> (p. 4164) ) . . . . .	4164
<b>activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>XATransactionId-</b> <b>Marshaller</b> (p. 4168) ) . . . . .	4168
<b>decaf::util::logging::XMLFormatter</b> (Format a <b>LogRecord</b> (p. 2487) into a standard XML format ) . . . . .	4172
<b>z_stream_s</b> . . . . .	4174
<b>decaf::util::zip::ZipException</b> . . . . .	4175

## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

src/main/activemq/cmsutil/ <b>CachedConsumer.h</b> . . . . .	4179
src/main/activemq/cmsutil/ <b>CachedProducer.h</b> . . . . .	4179
src/main/activemq/cmsutil/ <b>CmsAccessor.h</b> . . . . .	4180
src/main/activemq/cmsutil/ <b>CmsDestinationAccessor.h</b> . . . . .	4180
src/main/activemq/cmsutil/ <b>CmsTemplate.h</b> . . . . .	4181
src/main/activemq/cmsutil/ <b>DestinationResolver.h</b> . . . . .	4182
src/main/activemq/cmsutil/ <b>DynamicDestinationResolver.h</b> . . . . .	4182
src/main/activemq/cmsutil/ <b>MessageCreator.h</b> . . . . .	4183
src/main/activemq/cmsutil/ <b>PooledSession.h</b> . . . . .	4183
src/main/activemq/cmsutil/ <b>ProducerCallback.h</b> . . . . .	4184
src/main/activemq/cmsutil/ <b>ResourceLifecycleManager.h</b> . . . . .	4184
src/main/activemq/cmsutil/ <b>SessionCallback.h</b> . . . . .	4185
src/main/activemq/cmsutil/ <b>SessionPool.h</b> . . . . .	4186
src/main/activemq/commands/ <b>ActiveMQBlobMessage.h</b> . . . . .	4186
src/main/activemq/commands/ <b>ActiveMQBytesMessage.h</b> . . . . .	4187
src/main/activemq/commands/ <b>ActiveMQDestination.h</b> . . . . .	4188
src/main/activemq/commands/ <b>ActiveMQMapMessage.h</b> . . . . .	4188
src/main/activemq/commands/ <b>ActiveMQMessage.h</b> . . . . .	4189
src/main/activemq/commands/ <b>ActiveMQMessageTemplate.h</b> . . . . .	4189
src/main/activemq/commands/ <b>ActiveMQObjectMessage.h</b> . . . . .	4190
src/main/activemq/commands/ <b>ActiveMQQueue.h</b> . . . . .	4191
src/main/activemq/commands/ <b>ActiveMQStreamMessage.h</b> . . . . .	4191
src/main/activemq/commands/ <b>ActiveMQTempDestination.h</b> . . . . .	4192
src/main/activemq/commands/ <b>ActiveMQTempQueue.h</b> . . . . .	4193
src/main/activemq/commands/ <b>ActiveMQTempTopic.h</b> . . . . .	4193
src/main/activemq/commands/ <b>ActiveMQTextMessage.h</b> . . . . .	4194
src/main/activemq/commands/ <b>ActiveMQTopic.h</b> . . . . .	4194
src/main/activemq/commands/ <b>BaseCommand.h</b> . . . . .	4195
src/main/activemq/commands/ <b>BaseDataStructure.h</b> . . . . .	4195

src/main/activemq/commands/ <b>BooleanExpression.h</b>	4196
src/main/activemq/commands/ <b>BrokerError.h</b>	4196
src/main/activemq/commands/ <b>BrokerId.h</b>	4197
src/main/activemq/commands/ <b>BrokerInfo.h</b>	4197
src/main/activemq/commands/ <b>Command.h</b>	4198
src/main/activemq/commands/ <b>ConnectionControl.h</b>	4199
src/main/activemq/commands/ <b>ConnectionError.h</b>	4199
src/main/activemq/commands/ <b>ConnectionId.h</b>	4200
src/main/activemq/commands/ <b>ConnectionInfo.h</b>	4200
src/main/activemq/commands/ <b>ConsumerControl.h</b>	4201
src/main/activemq/commands/ <b>ConsumerId.h</b>	4201
src/main/activemq/commands/ <b>ConsumerInfo.h</b>	4202
src/main/activemq/commands/ <b>ControlCommand.h</b>	4203
src/main/activemq/commands/ <b>dataArrayResponse.h</b>	4203
src/main/activemq/commands/ <b>DataResponse.h</b>	4204
src/main/activemq/commands/ <b>DataStructure.h</b>	4204
src/main/activemq/commands/ <b>DestinationInfo.h</b>	4205
src/main/activemq/commands/ <b>DiscoveryEvent.h</b>	4205
src/main/activemq/commands/ <b>ExceptionResponse.h</b>	4206
src/main/activemq/commands/ <b>FlushCommand.h</b>	4206
src/main/activemq/commands/ <b>IntegerResponse.h</b>	4207
src/main/activemq/commands/ <b>JournalQueueAck.h</b>	4207
src/main/activemq/commands/ <b>JournalTopicAck.h</b>	4208
src/main/activemq/commands/ <b>JournalTrace.h</b>	4209
src/main/activemq/commands/ <b>JournalTransaction.h</b>	4209
src/main/activemq/commands/ <b>KeepAliveInfo.h</b>	4210
src/main/activemq/commands/ <b>LastPartialCommand.h</b>	4210
src/main/activemq/commands/ <b>LocalTransactionId.h</b>	4211
src/main/activemq/commands/ <b>Message.h</b>	4211
src/main/activemq/commands/ <b>MessageAck.h</b>	4213
src/main/activemq/commands/ <b>MessageDispatch.h</b>	4213
src/main/activemq/commands/ <b>MessageDispatchNotification.h</b>	4214
src/main/activemq/commands/ <b>MessageId.h</b>	4215
src/main/activemq/commands/ <b>MessagePull.h</b>	4215
src/main/activemq/commands/ <b>NetworkBridgeFilter.h</b>	4216
src/main/activemq/commands/ <b>PartialCommand.h</b>	4216
src/main/activemq/commands/ <b>ProducerAck.h</b>	4217
src/main/activemq/commands/ <b>ProducerId.h</b>	4217
src/main/activemq/commands/ <b>ProducerInfo.h</b>	4218
src/main/activemq/commands/ <b>RemoveInfo.h</b>	4219
src/main/activemq/commands/ <b>RemoveSubscriptionInfo.h</b>	4219
src/main/activemq/commands/ <b>ReplayCommand.h</b>	4220
src/main/activemq/commands/ <b>Response.h</b>	4220
src/main/activemq/commands/ <b>SessionId.h</b>	4221
src/main/activemq/commands/ <b>SessionInfo.h</b>	4221
src/main/activemq/commands/ <b>ShutdownInfo.h</b>	4222
src/main/activemq/commands/ <b>SubscriptionInfo.h</b>	4222
src/main/activemq/commands/ <b>TransactionId.h</b>	4223
src/main/activemq/commands/ <b>TransactionInfo.h</b>	4223
src/main/activemq/commands/ <b>WireFormatInfo.h</b>	4224

src/main/activemq/commands/XATransactionId.h	4224
src/main/activemq/core/ActiveMQAckHandler.h	4225
src/main/activemq/core/ActiveMQConnection.h	4225
src/main/activemq/core/ActiveMQConnectionFactory.h	4226
src/main/activemq/core/ActiveMQConnectionMetaData.h	4227
src/main/activemq/core/ActiveMQConstants.h	4227
src/main/activemq/core/ActiveMQConsumer.h	4228
src/main/activemq/core/ActiveMQProducer.h	4229
src/main/activemq/core/ActiveMQQueueBrowser.h	4230
src/main/activemq/core/ActiveMQSession.h	4230
src/main/activemq/core/ActiveMQSessionExecutor.h	4231
src/main/activemq/core/ActiveMQTransactionContext.h	4232
src/main/activemq/core/DispatchData.h	4233
src/main/activemq/core/Dispatcher.h	4233
src/main/activemq/core/MessageDispatchChannel.h	4234
src/main/activemq/core/PrefetchPolicy.h	4235
src/main/activemq/core/RedeliveryPolicy.h	4236
src/main/activemq/core/Synchronization.h	4236
src/main/activemq/core/policies/DefaultPrefetchPolicy.h	4234
src/main/activemq/core/policies/DefaultRedeliveryPolicy.h	4235
src/main/activemq/exceptions/ActiveMQException.h	4237
src/main/activemq/exceptions/BrokerException.h	4237
src/main/activemq/exceptions/ExceptionDefines.h	4238
src/main/activemq/io/LoggingInputStream.h	4242
src/main/activemq/io/LoggingOutputStream.h	4243
src/main/activemq/library/ActiveMQCPP.h	4243
src/main/activemq/state/CommandVisitor.h	4244
src/main/activemq/state/CommandVisitorAdapter.h	4244
src/main/activemq/state/ConnectionState.h	4245
src/main/activemq/state/ConnectionStateTracker.h	4246
src/main/activemq/state/ConsumerState.h	4247
src/main/activemq/state/ProducerState.h	4248
src/main/activemq/state/SessionState.h	4248
src/main/activemq/state/Tracked.h	4249
src/main/activemq/state/TransactionState.h	4249
src/main/activemq/threads/CompositeTask.h	4250
src/main/activemq/threads/CompositeTaskRunner.h	4250
src/main/activemq/threads/DedicatedTaskRunner.h	4251
src/main/activemq/threads/Task.h	4252
src/main/activemq/threads/TaskRunner.h	4252
src/main/activemq/transport/AbstractTransportFactory.h	4253
src/main/activemq/transport/CompositeTransport.h	4253
src/main/activemq/transport/DefaultTransportListener.h	4255
src/main/activemq/transport/IOTransport.h	4262
src/main/activemq/transport/Transport.h	4268
src/main/activemq/transport/TransportFactory.h	4269
src/main/activemq/transport/TransportFilter.h	4270
src/main/activemq/transport/TransportListener.h	4270
src/main/activemq/transport/TransportRegistry.h	4271
src/main/activemq/transport/correlator/FutureResponse.h	4254

src/main/activemq/transport/correlator/ <b>ResponseCorrelator.h</b> . . . . .	4254
src/main/activemq/transport/failover/ <b>BackupTransport.h</b> . . . . .	4256
src/main/activemq/transport/failover/ <b>BackupTransportPool.h</b> . . . . .	4256
src/main/activemq/transport/failover/ <b>CloseTransportsTask.h</b> . . . . .	4257
src/main/activemq/transport/failover/ <b>FailoverTransport.h</b> . . . . .	4257
src/main/activemq/transport/failover/ <b>FailoverTransportFactory.h</b> . . . . .	4258
src/main/activemq/transport/failover/ <b>FailoverTransportListener.h</b> . . . . .	4259
src/main/activemq/transport/failover/ <b>URIPool.h</b> . . . . .	4260
src/main/activemq/transport/inactivity/ <b>InactivityMonitor.h</b> . . . . .	4260
src/main/activemq/transport/inactivity/ <b>ReadChecker.h</b> . . . . .	4261
src/main/activemq/transport/inactivity/ <b>WriteChecker.h</b> . . . . .	4261
src/main/activemq/transport/logging/ <b>LoggingTransport.h</b> . . . . .	4263
src/main/activemq/transport/mock/ <b>InternalCommandListener.h</b> . . . . .	4263
src/main/activemq/transport/mock/ <b>MockTransport.h</b> . . . . .	4264
src/main/activemq/transport/mock/ <b>MockTransportFactory.h</b> . . . . .	4265
src/main/activemq/transport/mock/ <b>ResponseBuilder.h</b> . . . . .	4265
src/main/activemq/transport/tcp/ <b>SslTransport.h</b> . . . . .	4266
src/main/activemq/transport/tcp/ <b>SslTransportFactory.h</b> . . . . .	4266
src/main/activemq/transport/tcp/ <b>TcpTransport.h</b> . . . . .	4267
src/main/activemq/transport/tcp/ <b>TcpTransportFactory.h</b> . . . . .	4268
src/main/activemq/util/ <b>ActiveMQProperties.h</b> . . . . .	4271
src/main/activemq/util/ <b>CMSExceptionSupport.h</b> . . . . .	4272
src/main/activemq/util/ <b>CompositeData.h</b> . . . . .	4274
src/main/activemq/util/ <b>Config.h</b> . . . . .	4274
src/main/activemq/util/ <b>IdGenerator.h</b> . . . . .	4275
src/main/activemq/util/ <b>LongSequenceGenerator.h</b> . . . . .	4276
src/main/activemq/util/ <b>MarshallingSupport.h</b> . . . . .	4276
src/main/activemq/util/ <b>MemoryUsage.h</b> . . . . .	4277
src/main/activemq/util/ <b>PrimitiveList.h</b> . . . . .	4277
src/main/activemq/util/ <b>PrimitiveMap.h</b> . . . . .	4278
src/main/activemq/util/ <b>PrimitiveValueConverter.h</b> . . . . .	4279
src/main/activemq/util/ <b>PrimitiveValueNode.h</b> . . . . .	4279
src/main/activemq/util/ <b>URISupport.h</b> . . . . .	4280
src/main/activemq/util/ <b>Usage.h</b> . . . . .	4280
src/main/activemq/wireformat/ <b>MarshalAware.h</b> . . . . .	4281
src/main/activemq/wireformat/ <b>WireFormat.h</b> . . . . .	4569
src/main/activemq/wireformat/ <b>WireFormatFactory.h</b> . . . . .	4569
src/main/activemq/wireformat/ <b>WireFormatNegotiator.h</b> . . . . .	4570
src/main/activemq/wireformat/ <b>WireFormatRegistry.h</b> . . . . .	4570
src/main/activemq/wireformat/openwire/ <b>OpenWireFormat.h</b> . . . . .	4561
src/main/activemq/wireformat/openwire/ <b>OpenWireFormatFactory.h</b> . . . . .	4562
src/main/activemq/wireformat/openwire/ <b>OpenWireFormatNegotiator.h</b> . . . . .	4562
src/main/activemq/wireformat/openwire/ <b>OpenWireResponseBuilder.h</b> . . . . .	4563
src/main/activemq/wireformat/openwire/marshal/ <b>BaseDataStreamMarshaller.h</b> 4281	
src/main/activemq/wireformat/openwire/marshal/ <b>DataStreamMarshaller.h</b> . . . . .	4282
src/main/activemq/wireformat/openwire/marshal/ <b>PrimitiveTypesMarshaller.h</b> 4282	
src/main/activemq/wireformat/openwire/marshal/v1/ <b>ActiveMQBlobMessageMarshaller.h</b> 4283	

src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBytesMessageMarshaller.h  
4288  
src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h  
4292  
src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarshaller.h  
4297  
src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h  
4301  
src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMarshaller.h  
4306  
src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h  
4310  
src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQStreamMessageMarshaller.h  
4315  
src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h  
4319  
src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h  
4324  
src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h  
4329  
src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h  
4333  
src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h  
4338  
src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h  
4342  
src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdMarshaller.h 4347  
src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfoMarshaller.h  
4351  
src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h  
4355  
src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h  
4360  
src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h  
4364  
src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h  
4369  
src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller.h  
4373  
src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h  
4378  
src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h  
4382  
src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h  
4387  
src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h  
4391  
src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h  
4396

src/main/activemq/wireformat/openwire/marshal/v1/**DestinationInfoMarshaller.h**  
4400  
src/main/activemq/wireformat/openwire/marshal/v1/**DiscoveryEventMarshaller.h**  
4405  
src/main/activemq/wireformat/openwire/marshal/v1/**ExceptionResponseMarshaller.h**  
4409  
src/main/activemq/wireformat/openwire/marshal/v1/**FlushCommandMarshaller.h**  
4414  
src/main/activemq/wireformat/openwire/marshal/v1/**IntegerResponseMarshaller.h**  
4418  
src/main/activemq/wireformat/openwire/marshal/v1/**JournalQueueAckMarshaller.h**  
4423  
src/main/activemq/wireformat/openwire/marshal/v1/**JournalTopicAckMarshaller.h**  
4427  
src/main/activemq/wireformat/openwire/marshal/v1/**JournalTraceMarshaller.h**  
4432  
src/main/activemq/wireformat/openwire/marshal/v1/**JournalTransactionMarshaller.h**  
4436  
src/main/activemq/wireformat/openwire/marshal/v1/**KeepAliveInfoMarshaller.h**  
4441  
src/main/activemq/wireformat/openwire/marshal/v1/**LastPartialCommandMarshaller.h**  
4445  
src/main/activemq/wireformat/openwire/marshal/v1/**LocalTransactionIdMarshaller.h**  
4450  
src/main/activemq/wireformat/openwire/marshal/v1/**MarshallerFactory.h** . 4454  
src/main/activemq/wireformat/openwire/marshal/v1/**MessageAckMarshaller.h**  
4458  
src/main/activemq/wireformat/openwire/marshal/v1/**MessageDispatchMarshaller.h**  
4462  
src/main/activemq/wireformat/openwire/marshal/v1/**MessageDispatchNotificationMarshaller.h**  
4467  
src/main/activemq/wireformat/openwire/marshal/v1/**MessageIdMarshaller.h**  
4471  
src/main/activemq/wireformat/openwire/marshal/v1/**MessageMarshaller.h** . 4476  
src/main/activemq/wireformat/openwire/marshal/v1/**MessagePullMarshaller.h**  
4480  
src/main/activemq/wireformat/openwire/marshal/v1/**NetworkBridgeFilterMarshaller.h**  
4485  
src/main/activemq/wireformat/openwire/marshal/v1/**PartialCommandMarshaller.h**  
4489  
src/main/activemq/wireformat/openwire/marshal/v1/**ProducerAckMarshaller.h**  
4494  
src/main/activemq/wireformat/openwire/marshal/v1/**ProducerIdMarshaller.h**  
4498  
src/main/activemq/wireformat/openwire/marshal/v1/**ProducerInfoMarshaller.h**  
4503  
src/main/activemq/wireformat/openwire/marshal/v1/**RemoveInfoMarshaller.h**  
4507  
src/main/activemq/wireformat/openwire/marshal/v1/**RemoveSubscriptionInfoMarshaller.h**  
4512



src/main/activemq/wireformat/openwire/marshal/v1/**ReplayCommandMarshaller.h**  
4516  
src/main/activemq/wireformat/openwire/marshal/v1/**ResponseMarshaller.h**  
4521  
src/main/activemq/wireformat/openwire/marshal/v1/**SessionIdMarshaller.h**  
4525  
src/main/activemq/wireformat/openwire/marshal/v1/**SessionInfoMarshaller.h**  
4529  
src/main/activemq/wireformat/openwire/marshal/v1/**ShutdownInfoMarshaller.h**  
4534  
src/main/activemq/wireformat/openwire/marshal/v1/**SubscriptionInfoMarshaller.h**  
4538  
src/main/activemq/wireformat/openwire/marshal/v1/**TransactionIdMarshaller.h**  
4543  
src/main/activemq/wireformat/openwire/marshal/v1/**TransactionInfoMarshaller.h**  
4547  
src/main/activemq/wireformat/openwire/marshal/v1/**WireFormatInfoMarshaller.h**  
4552  
src/main/activemq/wireformat/openwire/marshal/v1/**XATransactionIdMarshaller.h**  
4556  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQBlobMessageMarshaller.h**  
4284  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQBytesMessageMarshaller.h**  
4288  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQDestinationMarshaller.h**  
4293  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQMapMessageMarshaller.h**  
4297  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQMessageMarshaller.h**  
4302  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQObjectMessageMarshaller.h**  
4306  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQQueueMarshaller.h**  
4311  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQStreamMessageMarshaller.h**  
4315  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQTempDestinationMarshaller.h**  
4320  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQTempQueueMarshaller.h**  
4325  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQTempTopicMarshaller.h**  
4329  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQTextMessageMarshaller.h**  
4334  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQTopicMarshaller.h**  
4338  
src/main/activemq/wireformat/openwire/marshal/v2/**BaseCommandMarshaller.h**  
4343  
src/main/activemq/wireformat/openwire/marshal/v2/**BrokerIdMarshaller.h** 4347

src/main/activemq/wireformat/openwire/marshal/v2/**BrokerInfoMarshaller.h**  
4352  
src/main/activemq/wireformat/openwire/marshal/v2/**ConnectionControlMarshaller.h**  
4356  
src/main/activemq/wireformat/openwire/marshal/v2/**ConnectionErrorMarshaller.h**  
4361  
src/main/activemq/wireformat/openwire/marshal/v2/**ConnectionIdMarshaller.h**  
4365  
src/main/activemq/wireformat/openwire/marshal/v2/**ConnectionInfoMarshaller.h**  
4370  
src/main/activemq/wireformat/openwire/marshal/v2/**ConsumerControlMarshaller.h**  
4374  
src/main/activemq/wireformat/openwire/marshal/v2/**ConsumerIdMarshaller.h**  
4379  
src/main/activemq/wireformat/openwire/marshal/v2/**ConsumerInfoMarshaller.h**  
4383  
src/main/activemq/wireformat/openwire/marshal/v2/**ControlCommandMarshaller.h**  
4388  
src/main/activemq/wireformat/openwire/marshal/v2/**DataArrayResponseMarshaller.h**  
4392  
src/main/activemq/wireformat/openwire/marshal/v2/**DataResponseMarshaller.h**  
4397  
src/main/activemq/wireformat/openwire/marshal/v2/**DestinationInfoMarshaller.h**  
4401  
src/main/activemq/wireformat/openwire/marshal/v2/**DiscoveryEventMarshaller.h**  
4406  
src/main/activemq/wireformat/openwire/marshal/v2/**ExceptionResponseMarshaller.h**  
4410  
src/main/activemq/wireformat/openwire/marshal/v2/**FlushCommandMarshaller.h**  
4415  
src/main/activemq/wireformat/openwire/marshal/v2/**IntegerResponseMarshaller.h**  
4419  
src/main/activemq/wireformat/openwire/marshal/v2/**JournalQueueAckMarshaller.h**  
4424  
src/main/activemq/wireformat/openwire/marshal/v2/**JournalTopicAckMarshaller.h**  
4428  
src/main/activemq/wireformat/openwire/marshal/v2/**JournalTraceMarshaller.h**  
4433  
src/main/activemq/wireformat/openwire/marshal/v2/**JournalTransactionMarshaller.h**  
4437  
src/main/activemq/wireformat/openwire/marshal/v2/**KeepAliveInfoMarshaller.h**  
4442  
src/main/activemq/wireformat/openwire/marshal/v2/**LastPartialCommandMarshaller.h**  
4446  
src/main/activemq/wireformat/openwire/marshal/v2/**LocalTransactionIdMarshaller.h**  
4451  
src/main/activemq/wireformat/openwire/marshal/v2/**MarshallerFactory.h** . 4455  
src/main/activemq/wireformat/openwire/marshal/v2/**MessageAckMarshaller.h**  
4458

src/main/activemq/wireformat/openwire/marshal/v2/**MessageDispatchMarshaller.h**  
4463

src/main/activemq/wireformat/openwire/marshal/v2/**MessageDispatchNotificationMarshaller.h**  
4467

src/main/activemq/wireformat/openwire/marshal/v2/**MessageIdMarshaller.h**  
4472

src/main/activemq/wireformat/openwire/marshal/v2/**MessageMarshaller.h** . 4476

src/main/activemq/wireformat/openwire/marshal/v2/**MessagePullMarshaller.h**  
4481

src/main/activemq/wireformat/openwire/marshal/v2/**NetworkBridgeFilterMarshaller.h**  
4485

src/main/activemq/wireformat/openwire/marshal/v2/**PartialCommandMarshaller.h**  
4490

src/main/activemq/wireformat/openwire/marshal/v2/**ProducerAckMarshaller.h**  
4494

src/main/activemq/wireformat/openwire/marshal/v2/**ProducerIdMarshaller.h**  
4499

src/main/activemq/wireformat/openwire/marshal/v2/**ProducerInfoMarshaller.h**  
4503

src/main/activemq/wireformat/openwire/marshal/v2/**RemoveInfoMarshaller.h**  
4508

src/main/activemq/wireformat/openwire/marshal/v2/**RemoveSubscriptionInfoMarshaller.h**  
4512

src/main/activemq/wireformat/openwire/marshal/v2/**ReplayCommandMarshaller.h**  
4517

src/main/activemq/wireformat/openwire/marshal/v2/**ResponseMarshaller.h**  
4521

src/main/activemq/wireformat/openwire/marshal/v2/**SessionIdMarshaller.h**  
4526

src/main/activemq/wireformat/openwire/marshal/v2/**SessionInfoMarshaller.h**  
4530

src/main/activemq/wireformat/openwire/marshal/v2/**ShutdownInfoMarshaller.h**  
4535

src/main/activemq/wireformat/openwire/marshal/v2/**SubscriptionInfoMarshaller.h**  
4539

src/main/activemq/wireformat/openwire/marshal/v2/**TransactionIdMarshaller.h**  
4544

src/main/activemq/wireformat/openwire/marshal/v2/**TransactionInfoMarshaller.h**  
4548

src/main/activemq/wireformat/openwire/marshal/v2/**WireFormatInfoMarshaller.h**  
4553

src/main/activemq/wireformat/openwire/marshal/v2/**XATransactionIdMarshaller.h**  
4557

src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQBlobMessageMarshaller.h**  
4285

src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQBytesMessageMarshaller.h**  
4289

src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQDestinationMarshaller.h**  
4294

src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQMapMessageMarshaller.h**  
4298  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQMessageMarshaller.h**  
4303  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQObjectMessageMarshaller.h**  
4307  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQQueueMarshaller.h**  
4312  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQStreamMessageMarshaller.h**  
4316  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTempDestinationMarshaller.h**  
4321  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTempQueueMarshaller.h**  
4326  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTempTopicMarshaller.h**  
4330  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTextMessageMarshaller.h**  
4335  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTopicMarshaller.h**  
4339  
src/main/activemq/wireformat/openwire/marshal/v3/**BaseCommandMarshaller.h**  
4344  
src/main/activemq/wireformat/openwire/marshal/v3/**BrokerIdMarshaller.h** 4348  
src/main/activemq/wireformat/openwire/marshal/v3/**BrokerInfoMarshaller.h**  
4352  
src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionControlMarshaller.h**  
4357  
src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionErrorMarshaller.h**  
4361  
src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionIdMarshaller.h**  
4366  
src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionInfoMarshaller.h**  
4370  
src/main/activemq/wireformat/openwire/marshal/v3/**ConsumerControlMarshaller.h**  
4375  
src/main/activemq/wireformat/openwire/marshal/v3/**ConsumerIdMarshaller.h**  
4379  
src/main/activemq/wireformat/openwire/marshal/v3/**ConsumerInfoMarshaller.h**  
4384  
src/main/activemq/wireformat/openwire/marshal/v3/**ControlCommandMarshaller.h**  
4388  
src/main/activemq/wireformat/openwire/marshal/v3/**DataArrayResponseMarshaller.h**  
4393  
src/main/activemq/wireformat/openwire/marshal/v3/**DataResponseMarshaller.h**  
4397  
src/main/activemq/wireformat/openwire/marshal/v3/**DestinationInfoMarshaller.h**  
4402  
src/main/activemq/wireformat/openwire/marshal/v3/**DiscoveryEventMarshaller.h**  
4406

src/main/activemq/wireformat/openwire/marshall/v3/**ExceptionResponseMarshaller.h**  
4411  
src/main/activemq/wireformat/openwire/marshall/v3/**FlushCommandMarshaller.h**  
4415  
src/main/activemq/wireformat/openwire/marshall/v3/**IntegerResponseMarshaller.h**  
4420  
src/main/activemq/wireformat/openwire/marshall/v3/**JournalQueueAckMarshaller.h**  
4424  
src/main/activemq/wireformat/openwire/marshall/v3/**JournalTopicAckMarshaller.h**  
4429  
src/main/activemq/wireformat/openwire/marshall/v3/**JournalTraceMarshaller.h**  
4433  
src/main/activemq/wireformat/openwire/marshall/v3/**JournalTransactionMarshaller.h**  
4438  
src/main/activemq/wireformat/openwire/marshall/v3/**KeepAliveInfoMarshaller.h**  
4442  
src/main/activemq/wireformat/openwire/marshall/v3/**LastPartialCommandMarshaller.h**  
4447  
src/main/activemq/wireformat/openwire/marshall/v3/**LocalTransactionIdMarshaller.h**  
4451  
src/main/activemq/wireformat/openwire/marshall/v3/**MarshallerFactory.h** . 4455  
src/main/activemq/wireformat/openwire/marshall/v3/**MessageAckMarshaller.h**  
4459  
src/main/activemq/wireformat/openwire/marshall/v3/**MessageDispatchMarshaller.h**  
4464  
src/main/activemq/wireformat/openwire/marshall/v3/**MessageDispatchNotificationMarshaller.h**  
4468  
src/main/activemq/wireformat/openwire/marshall/v3/**MessageIdMarshaller.h**  
4473  
src/main/activemq/wireformat/openwire/marshall/v3/**MessageMarshaller.h** . 4477  
src/main/activemq/wireformat/openwire/marshall/v3/**MessagePullMarshaller.h**  
4482  
src/main/activemq/wireformat/openwire/marshall/v3/**NetworkBridgeFilterMarshaller.h**  
4486  
src/main/activemq/wireformat/openwire/marshall/v3/**PartialCommandMarshaller.h**  
4491  
src/main/activemq/wireformat/openwire/marshall/v3/**ProducerAckMarshaller.h**  
4495  
src/main/activemq/wireformat/openwire/marshall/v3/**ProducerIdMarshaller.h**  
4500  
src/main/activemq/wireformat/openwire/marshall/v3/**ProducerInfoMarshaller.h**  
4504  
src/main/activemq/wireformat/openwire/marshall/v3/**RemoveInfoMarshaller.h**  
4509  
src/main/activemq/wireformat/openwire/marshall/v3/**RemoveSubscriptionInfoMarshaller.h**  
4513  
src/main/activemq/wireformat/openwire/marshall/v3/**ReplayCommandMarshaller.h**  
4518  
src/main/activemq/wireformat/openwire/marshall/v3/**ResponseMarshaller.h**  
4522

src/main/activemq/wireformat/openwire/marshal/v3/**SessionIdMarshaller.h**  
4526  
src/main/activemq/wireformat/openwire/marshal/v3/**SessionInfoMarshaller.h**  
4531  
src/main/activemq/wireformat/openwire/marshal/v3/**ShutdownInfoMarshaller.h**  
4535  
src/main/activemq/wireformat/openwire/marshal/v3/**SubscriptionInfoMarshaller.h**  
4540  
src/main/activemq/wireformat/openwire/marshal/v3/**TransactionIdMarshaller.h**  
4544  
src/main/activemq/wireformat/openwire/marshal/v3/**TransactionInfoMarshaller.h**  
4549  
src/main/activemq/wireformat/openwire/marshal/v3/**WireFormatInfoMarshaller.h**  
4553  
src/main/activemq/wireformat/openwire/marshal/v3/**XATransactionIdMarshaller.h**  
4558  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQBlobMessageMarshaller.h**  
4285  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQBytesMessageMarshaller.h**  
4290  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQDestinationMarshaller.h**  
4294  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQMapMessageMarshaller.h**  
4299  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQMessageMarshaller.h**  
4303  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQObjectMessageMarshaller.h**  
4308  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQQueueMarshaller.h**  
4312  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQStreamMessageMarshaller.h**  
4317  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTempDestinationMarshaller.h**  
4322  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTempQueueMarshaller.h**  
4326  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTempTopicMarshaller.h**  
4331  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTextMessageMarshaller.h**  
4335  
src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTopicMarshaller.h**  
4340  
src/main/activemq/wireformat/openwire/marshal/v4/**BaseCommandMarshaller.h**  
4344  
src/main/activemq/wireformat/openwire/marshal/v4/**BrokerIdMarshaller.h** 4349  
src/main/activemq/wireformat/openwire/marshal/v4/**BrokerInfoMarshaller.h**  
4353  
src/main/activemq/wireformat/openwire/marshal/v4/**ConnectionControlMarshaller.h**  
4358

src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h  
4362  
src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h  
4367  
src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h  
4371  
src/main/activemq/wireformat/openwire/marshal/v4/ConsumerControlMarshaller.h  
4376  
src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h  
4380  
src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h  
4385  
src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h  
4389  
src/main/activemq/wireformat/openwire/marshal/v4/DataArrayResponseMarshaller.h  
4394  
src/main/activemq/wireformat/openwire/marshal/v4/DataResponseMarshaller.h  
4398  
src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller.h  
4403  
src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller.h  
4407  
src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h  
4412  
src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h  
4416  
src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h  
4421  
src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller.h  
4425  
src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h  
4430  
src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h  
4434  
src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarshaller.h  
4439  
src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h  
4443  
src/main/activemq/wireformat/openwire/marshal/v4/LastPartialCommandMarshaller.h  
4448  
src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarshaller.h  
4452  
src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h . 4456  
src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h  
4460  
src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h  
4464  
src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller.h  
4469

src/main/activemq/wireformat/openwire/marshal/v4/**MessageIdMarshaller.h**  
4473

src/main/activemq/wireformat/openwire/marshal/v4/**MessageMarshaller.h** . 4478

src/main/activemq/wireformat/openwire/marshal/v4/**MessagePullMarshaller.h**  
4482

src/main/activemq/wireformat/openwire/marshal/v4/**NetworkBridgeFilterMarshaller.h**  
4487

src/main/activemq/wireformat/openwire/marshal/v4/**PartialCommandMarshaller.h**  
4491

src/main/activemq/wireformat/openwire/marshal/v4/**ProducerAckMarshaller.h**  
4496

src/main/activemq/wireformat/openwire/marshal/v4/**ProducerIdMarshaller.h**  
4500

src/main/activemq/wireformat/openwire/marshal/v4/**ProducerInfoMarshaller.h**  
4505

src/main/activemq/wireformat/openwire/marshal/v4/**RemoveInfoMarshaller.h**  
4509

src/main/activemq/wireformat/openwire/marshal/v4/**RemoveSubscriptionInfoMarshaller.h**  
4514

src/main/activemq/wireformat/openwire/marshal/v4/**ReplayCommandMarshaller.h**  
4518

src/main/activemq/wireformat/openwire/marshal/v4/**ResponseMarshaller.h**  
4523

src/main/activemq/wireformat/openwire/marshal/v4/**SessionIdMarshaller.h**  
4527

src/main/activemq/wireformat/openwire/marshal/v4/**SessionInfoMarshaller.h**  
4532

src/main/activemq/wireformat/openwire/marshal/v4/**ShutdownInfoMarshaller.h**  
4536

src/main/activemq/wireformat/openwire/marshal/v4/**SubscriptionInfoMarshaller.h**  
4541

src/main/activemq/wireformat/openwire/marshal/v4/**TransactionIdMarshaller.h**  
4545

src/main/activemq/wireformat/openwire/marshal/v4/**TransactionInfoMarshaller.h**  
4550

src/main/activemq/wireformat/openwire/marshal/v4/**WireFormatInfoMarshaller.h**  
4554

src/main/activemq/wireformat/openwire/marshal/v4/**XATransactionIdMarshaller.h**  
4559

src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQBlobMessageMarshaller.h**  
4286

src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQBytesMessageMarshaller.h**  
4291

src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQDestinationMarshaller.h**  
4295

src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQMapMessageMarshaller.h**  
4300

src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQMessageMarshaller.h**  
4304



src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMarshaller.h  
4309

src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h  
4313

src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQStreamMessageMarshaller.h  
4318

src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h  
4322

src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarshaller.h  
4327

src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller.h  
4332

src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTextMessageMarshaller.h  
4336

src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h  
4341

src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h  
4345

src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdMarshaller.h 4349

src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfoMarshaller.h  
4354

src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h  
4358

src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h  
4363

src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h  
4367

src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h  
4372

src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h  
4376

src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h  
4381

src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h  
4385

src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h  
4390

src/main/activemq/wireformat/openwire/marshal/v5/DataArrayResponseMarshaller.h  
4394

src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h  
4399

src/main/activemq/wireformat/openwire/marshal/v5/DestinationInfoMarshaller.h  
4403

src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h  
4408

src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h  
4412

src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h  
4417

src/main/activemq/wireformat/openwire/marshal/v5/**IntegerResponseMarshaller.h**  
 4421  
 src/main/activemq/wireformat/openwire/marshal/v5/**JournalQueueAckMarshaller.h**  
 4426  
 src/main/activemq/wireformat/openwire/marshal/v5/**JournalTopicAckMarshaller.h**  
 4430  
 src/main/activemq/wireformat/openwire/marshal/v5/**JournalTraceMarshaller.h**  
 4435  
 src/main/activemq/wireformat/openwire/marshal/v5/**JournalTransactionMarshaller.h**  
 4439  
 src/main/activemq/wireformat/openwire/marshal/v5/**KeepAliveInfoMarshaller.h**  
 4444  
 src/main/activemq/wireformat/openwire/marshal/v5/**LastPartialCommandMarshaller.h**  
 4448  
 src/main/activemq/wireformat/openwire/marshal/v5/**LocalTransactionIdMarshaller.h**  
 4453  
 src/main/activemq/wireformat/openwire/marshal/v5/**MarshallerFactory.h** . 4456  
 src/main/activemq/wireformat/openwire/marshal/v5/**MessageAckMarshaller.h**  
 4461  
 src/main/activemq/wireformat/openwire/marshal/v5/**MessageDispatchMarshaller.h**  
 4465  
 src/main/activemq/wireformat/openwire/marshal/v5/**MessageDispatchNotificationMarshaller.h**  
 4470  
 src/main/activemq/wireformat/openwire/marshal/v5/**MessageIdMarshaller.h**  
 4474  
 src/main/activemq/wireformat/openwire/marshal/v5/**MessageMarshaller.h** . 4479  
 src/main/activemq/wireformat/openwire/marshal/v5/**MessagePullMarshaller.h**  
 4483  
 src/main/activemq/wireformat/openwire/marshal/v5/**NetworkBridgeFilterMarshaller.h**  
 4488  
 src/main/activemq/wireformat/openwire/marshal/v5/**PartialCommandMarshaller.h**  
 4492  
 src/main/activemq/wireformat/openwire/marshal/v5/**ProducerAckMarshaller.h**  
 4497  
 src/main/activemq/wireformat/openwire/marshal/v5/**ProducerIdMarshaller.h**  
 4501  
 src/main/activemq/wireformat/openwire/marshal/v5/**ProducerInfoMarshaller.h**  
 4506  
 src/main/activemq/wireformat/openwire/marshal/v5/**RemoveInfoMarshaller.h**  
 4510  
 src/main/activemq/wireformat/openwire/marshal/v5/**RemoveSubscriptionInfoMarshaller.h**  
 4515  
 src/main/activemq/wireformat/openwire/marshal/v5/**ReplayCommandMarshaller.h**  
 4519  
 src/main/activemq/wireformat/openwire/marshal/v5/**ResponseMarshaller.h**  
 4524  
 src/main/activemq/wireformat/openwire/marshal/v5/**SessionIdMarshaller.h**  
 4528  
 src/main/activemq/wireformat/openwire/marshal/v5/**SessionInfoMarshaller.h**  
 4532

src/main/activemq/wireformat/openwire/marshal/v5/**ShutdownInfoMarshaller.h**  
4537

src/main/activemq/wireformat/openwire/marshal/v5/**SubscriptionInfoMarshaller.h**  
4541

src/main/activemq/wireformat/openwire/marshal/v5/**TransactionIdMarshaller.h**  
4546

src/main/activemq/wireformat/openwire/marshal/v5/**TransactionInfoMarshaller.h**  
4550

src/main/activemq/wireformat/openwire/marshal/v5/**WireFormatInfoMarshaller.h**  
4555

src/main/activemq/wireformat/openwire/marshal/v5/**XATransactionIdMarshaller.h**  
4559

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQBlobMessageMarshaller.h**  
4287

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQBytesMessageMarshaller.h**  
4291

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQDestinationMarshaller.h**  
4296

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQMapMessageMarshaller.h**  
4300

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQMessageMarshaller.h**  
4305

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQObjectMessageMarshaller.h**  
4309

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQQueueMarshaller.h**  
4314

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQStreamMessageMarshaller.h**  
4319

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQTempDestinationMarshaller.h**  
4323

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQTempQueueMarshaller.h**  
4328

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQTempTopicMarshaller.h**  
4332

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQTextMessageMarshaller.h**  
4337

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQTopicMarshaller.h**  
4341

src/main/activemq/wireformat/openwire/marshal/v6/**BaseCommandMarshaller.h**  
4346

src/main/activemq/wireformat/openwire/marshal/v6/**BrokerIdMarshaller.h** 4350

src/main/activemq/wireformat/openwire/marshal/v6/**BrokerInfoMarshaller.h**  
4355

src/main/activemq/wireformat/openwire/marshal/v6/**ConnectionControlMarshaller.h**  
4359

src/main/activemq/wireformat/openwire/marshal/v6/**ConnectionErrorMarshaller.h**  
4364

src/main/activemq/wireformat/openwire/marshal/v6/**ConnectionIdMarshaller.h**  
4368

src/main/activemq/wireformat/openwire/marshall/v6/**ConnectionInfoMarshaller.h**  
4373  
src/main/activemq/wireformat/openwire/marshall/v6/**ConsumerControlMarshaller.h**  
4377  
src/main/activemq/wireformat/openwire/marshall/v6/**ConsumerIdMarshaller.h**  
4382  
src/main/activemq/wireformat/openwire/marshall/v6/**ConsumerInfoMarshaller.h**  
4386  
src/main/activemq/wireformat/openwire/marshall/v6/**ControlCommandMarshaller.h**  
4391  
src/main/activemq/wireformat/openwire/marshall/v6/**DataArrayResponseMarshaller.h**  
4395  
src/main/activemq/wireformat/openwire/marshall/v6/**DataResponseMarshaller.h**  
4400  
src/main/activemq/wireformat/openwire/marshall/v6/**DestinationInfoMarshaller.h**  
4404  
src/main/activemq/wireformat/openwire/marshall/v6/**DiscoveryEventMarshaller.h**  
4409  
src/main/activemq/wireformat/openwire/marshall/v6/**ExceptionResponseMarshaller.h**  
4413  
src/main/activemq/wireformat/openwire/marshall/v6/**FlushCommandMarshaller.h**  
4418  
src/main/activemq/wireformat/openwire/marshall/v6/**IntegerResponseMarshaller.h**  
4422  
src/main/activemq/wireformat/openwire/marshall/v6/**JournalQueueAckMarshaller.h**  
4427  
src/main/activemq/wireformat/openwire/marshall/v6/**JournalTopicAckMarshaller.h**  
4431  
src/main/activemq/wireformat/openwire/marshall/v6/**JournalTraceMarshaller.h**  
4436  
src/main/activemq/wireformat/openwire/marshall/v6/**JournalTransactionMarshaller.h**  
4440  
src/main/activemq/wireformat/openwire/marshall/v6/**KeepAliveInfoMarshaller.h**  
4445  
src/main/activemq/wireformat/openwire/marshall/v6/**LastPartialCommandMarshaller.h**  
4449  
src/main/activemq/wireformat/openwire/marshall/v6/**LocalTransactionIdMarshaller.h**  
4454  
src/main/activemq/wireformat/openwire/marshall/v6/**MarshallerFactory.h** . 4457  
src/main/activemq/wireformat/openwire/marshall/v6/**MessageAckMarshaller.h**  
4461  
src/main/activemq/wireformat/openwire/marshall/v6/**MessageDispatchMarshaller.h**  
4466  
src/main/activemq/wireformat/openwire/marshall/v6/**MessageDispatchNotificationMarshaller.h**  
4470  
src/main/activemq/wireformat/openwire/marshall/v6/**MessageIdMarshaller.h**  
4475  
src/main/activemq/wireformat/openwire/marshall/v6/**MessageMarshaller.h** . 4479  
src/main/activemq/wireformat/openwire/marshall/v6/**MessagePullMarshaller.h**  
4484

src/main/activemq/wireformat/openwire/marshal/v6/ <b>NetworkBridgeFilterMarshaller.h</b>	
4488	
src/main/activemq/wireformat/openwire/marshal/v6/ <b>PartialCommandMarshaller.h</b>	
4493	
src/main/activemq/wireformat/openwire/marshal/v6/ <b>ProducerAckMarshaller.h</b>	
4497	
src/main/activemq/wireformat/openwire/marshal/v6/ <b>ProducerIdMarshaller.h</b>	
4502	
src/main/activemq/wireformat/openwire/marshal/v6/ <b>ProducerInfoMarshaller.h</b>	
4506	
src/main/activemq/wireformat/openwire/marshal/v6/ <b>RemoveInfoMarshaller.h</b>	
4511	
src/main/activemq/wireformat/openwire/marshal/v6/ <b>RemoveSubscriptionInfoMarshaller.h</b>	
4515	
src/main/activemq/wireformat/openwire/marshal/v6/ <b>ReplayCommandMarshaller.h</b>	
4520	
src/main/activemq/wireformat/openwire/marshal/v6/ <b>ResponseMarshaller.h</b>	
4524	
src/main/activemq/wireformat/openwire/marshal/v6/ <b>SessionIdMarshaller.h</b>	
4529	
src/main/activemq/wireformat/openwire/marshal/v6/ <b>SessionInfoMarshaller.h</b>	
4533	
src/main/activemq/wireformat/openwire/marshal/v6/ <b>ShutdownInfoMarshaller.h</b>	
4538	
src/main/activemq/wireformat/openwire/marshal/v6/ <b>SubscriptionInfoMarshaller.h</b>	
4542	
src/main/activemq/wireformat/openwire/marshal/v6/ <b>TransactionIdMarshaller.h</b>	
4547	
src/main/activemq/wireformat/openwire/marshal/v6/ <b>TransactionInfoMarshaller.h</b>	
4551	
src/main/activemq/wireformat/openwire/marshal/v6/ <b>WireFormatInfoMarshaller.h</b>	
4556	
src/main/activemq/wireformat/openwire/marshal/v6/ <b>XATransactionIdMarshaller.h</b>	
4560	
src/main/activemq/wireformat/openwire/utis/ <b>BooleanStream.h</b> . . . . .	4564
src/main/activemq/wireformat/openwire/utis/ <b>HexTable.h</b> . . . . .	4564
src/main/activemq/wireformat/openwire/utis/ <b>MessagePropertyInterceptor.h</b>	
4565	
src/main/activemq/wireformat/stomp/ <b>StompCommandConstants.h</b> . . . . .	4565
src/main/activemq/wireformat/stomp/ <b>StompFrame.h</b> . . . . .	4566
src/main/activemq/wireformat/stomp/ <b>StompHelper.h</b> . . . . .	4567
src/main/activemq/wireformat/stomp/ <b>StompWireFormat.h</b> . . . . .	4567
src/main/activemq/wireformat/stomp/ <b>StompWireFormatFactory.h</b> . . . . .	4568
src/main/cms/ <b>BytesMessage.h</b> . . . . .	4571
src/main/cms/ <b>Closeable.h</b> . . . . .	4572
src/main/cms/ <b>CMSException.h</b> . . . . .	4573
src/main/cms/ <b>CMSProperties.h</b> . . . . .	4573
src/main/cms/ <b>CMSSecurityException.h</b> . . . . .	4574
src/main/cms/ <b>Config.h</b> . . . . .	4275
src/main/cms/ <b>Connection.h</b> . . . . .	4574

src/main/cms/ConnectionFactory.h	4575
src/main/cms/ConnectionMetaData.h	4575
src/main/cms/DeliveryMode.h	4576
src/main/cms/Destination.h	4576
src/main/cms/ExceptionListener.h	4576
src/main/cms/IllegalStateException.h	4577
src/main/cms/InvalidClientIdException.h	4578
src/main/cms/InvalidDestinationException.h	4578
src/main/cms/InvalidSelectorException.h	4579
src/main/cms/MapMessage.h	4579
src/main/cms/Message.h	4212
src/main/cms/MessageConsumer.h	4580
src/main/cms/MessageEnumeration.h	4580
src/main/cms/MessageEOFException.h	4581
src/main/cms/MessageFormatException.h	4581
src/main/cms/MessageListener.h	4582
src/main/cms/MessageNotReadableException.h	4582
src/main/cms/MessageNotWriteableException.h	4583
src/main/cms/MessageProducer.h	4583
src/main/cms/ObjectMessage.h	4584
src/main/cms/Queue.h	4584
src/main/cms/QueueBrowser.h	4585
src/main/cms/Session.h	4586
src/main/cms/Startable.h	4587
src/main/cms/Stopable.h	4587
src/main/cms/StreamMessage.h	4587
src/main/cms/TemporaryQueue.h	4588
src/main/cms/TemporaryTopic.h	4588
src/main/cms/TextMessage.h	4589
src/main/cms/Topic.h	4589
src/main/cms/UnsupportedOperationException.h	4590
src/main/decaf/internal/AprPool.h	4591
src/main/decaf/internal/DecafRuntime.h	4591
src/main/decaf/internal/io/StandardErrorOutputStream.h	4592
src/main/decaf/internal/io/StandardInputStream.h	4592
src/main/decaf/internal/io/StandardOutputStream.h	4593
src/main/decaf/internal/net/DefaultServerSocketFactory.h	4593
src/main/decaf/internal/net/DefaultSocketFactory.h	4594
src/main/decaf/internal/net/Network.h	4594
src/main/decaf/internal/net/SocketFileDescriptor.h	4595
src/main/decaf/internal/net/URIEncoderDecoder.h	4604
src/main/decaf/internal/net/URIHelper.h	4605
src/main/decaf/internal/net/URIType.h	4605
src/main/decaf/internal/net/ssl/DefaultSSLContext.h	4595
src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h	4596
src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h	4597
src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h	4597
src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h	4598
src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h	4598
src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h	4599

src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h . . . . .	4600
src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h . . . .	4600
src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h . . . . .	4601
src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h . . .	4601
src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h . .	4602
src/main/decaf/internal/net/tcp/TcpSocket.h . . . . .	4603
src/main/decaf/internal/net/tcp/TcpSocketInputStream.h . . . . .	4603
src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h . . . . .	4604
src/main/decaf/internal/nio/BufferFactory.h . . . . .	4606
src/main/decaf/internal/nio/ByteArrayBuffer.h . . . . .	4607
src/main/decaf/internal/nio/CharArrayBuffer.h . . . . .	4607
src/main/decaf/internal/nio/DoubleArrayBuffer.h . . . . .	4608
src/main/decaf/internal/nio/FloatArrayBuffer.h . . . . .	4609
src/main/decaf/internal/nio/IntArrayBuffer.h . . . . .	4609
src/main/decaf/internal/nio/LongArrayBuffer.h . . . . .	4610
src/main/decaf/internal/nio/ShortArrayBuffer.h . . . . .	4610
src/main/decaf/internal/security/unix/SecureRandomImpl.h . . . . .	4611
src/main/decaf/internal/security/windows/SecureRandomImpl.h . . . . .	4612
src/main/decaf/internal/util/ByteArrayAdapter.h . . . . .	4612
src/main/decaf/internal/util/GenericResource.h . . . . .	4618
src/main/decaf/internal/util/HexStringParser.h . . . . .	4618
src/main/decaf/internal/util/Resource.h . . . . .	4619
src/main/decaf/internal/util/ResourceLifecycleManager.h . . . . .	4185
src/main/decaf/internal/util/TimerTaskHeap.h . . . . .	4619
src/main/decaf/internal/util/concurrent/ConditionImpl.h . . . . .	4613
src/main/decaf/internal/util/concurrent/MutexImpl.h . . . . .	4613
src/main/decaf/internal/util/concurrent/SynchronizableImpl.h . . . . .	4614
src/main/decaf/internal/util/concurrent/Transferer.h . . . . .	4614
src/main/decaf/internal/util/concurrent/TransferQueue.h . . . . .	4615
src/main/decaf/internal/util/concurrent/TransferStack.h . . . . .	4616
src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h . . . . .	4616
src/main/decaf/internal/util/concurrent/unix/MutexHandle.h . . . . .	4617
src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h . . . .	4617
src/main/decaf/internal/util/concurrent/windows/MutexHandle.h . . . . .	4617
src/main/decaf/internal/util/zip/crc32.h . . . . .	4620
src/main/decaf/internal/util/zip/deflate.h . . . . .	4620
src/main/decaf/internal/util/zip/gzguts.h . . . . .	4623
src/main/decaf/internal/util/zip/inffast.h . . . . .	4625
src/main/decaf/internal/util/zip/inffixed.h . . . . .	4626
src/main/decaf/internal/util/zip/inflate.h . . . . .	4626
src/main/decaf/internal/util/zip/inftrees.h . . . . .	4627
src/main/decaf/internal/util/zip/trees.h . . . . .	4628
src/main/decaf/internal/util/zip/zconf.h . . . . .	4630
src/main/decaf/internal/util/zip/zlib.h . . . . .	4632
src/main/decaf/internal/util/zip/zutil.h . . . . .	4636
src/main/decaf/io/BlockingByteArrayInputStream.h . . . . .	4639
src/main/decaf/io/BufferedInputStream.h . . . . .	4640
src/main/decaf/io/BufferedOutputStream.h . . . . .	4640
src/main/decaf/io/ByteArrayInputStream.h . . . . .	4641
src/main/decaf/io/ByteArrayOutputStream.h . . . . .	4641

src/main/decaf/io/Closeable.h	4572
src/main/decaf/io/DataInput.h	4642
src/main/decaf/io/DataInputStream.h	4642
src/main/decaf/io/DataOutput.h	4643
src/main/decaf/io/DataOutputStream.h	4644
src/main/decaf/io/EOFException.h	4644
src/main/decaf/io/FileDescriptor.h	4645
src/main/decaf/io/FilterInputStream.h	4645
src/main/decaf/io/FilterOutputStream.h	4646
src/main/decaf/io/Flushable.h	4646
src/main/decaf/io/InputStream.h	4647
src/main/decaf/io/InputStreamReader.h	4647
src/main/decaf/io/InterruptedIOException.h	4648
src/main/decaf/io/IOException.h	4648
src/main/decaf/io/OutputStream.h	4649
src/main/decaf/io/OutputStreamWriter.h	4649
src/main/decaf/io/PushbackInputStream.h	4650
src/main/decaf/io/Reader.h	4650
src/main/decaf/io/UnsupportedEncodingException.h	4651
src/main/decaf/io/UTFDataFormatException.h	4651
src/main/decaf/io/Writer.h	4652
src/main/decaf/lang/Appendable.h	4652
src/main/decaf/lang/ArrayPointer.h	4653
src/main/decaf/lang/Boolean.h	4654
src/main/decaf/lang/Byte.h	4654
src/main/decaf/lang/Character.h	4655
src/main/decaf/lang/CharSequence.h	4655
src/main/decaf/lang/Comparable.h	4656
src/main/decaf/lang/Double.h	4656
src/main/decaf/lang/Exception.h	4657
src/main/decaf/lang/Float.h	4662
src/main/decaf/lang/Integer.h	4663
src/main/decaf/lang/Iterable.h	4663
src/main/decaf/lang/Long.h	4664
src/main/decaf/lang/Math.h	4664
src/main/decaf/lang/Number.h	4664
src/main/decaf/lang/Pointer.h	4665
src/main/decaf/lang/Readable.h	4666
src/main/decaf/lang/Runnable.h	4667
src/main/decaf/lang/Runtime.h	4667
src/main/decaf/lang/Short.h	4667
src/main/decaf/lang/String.h	4668
src/main/decaf/lang/System.h	4668
src/main/decaf/lang/Thread.h	4669
src/main/decaf/lang/ThreadGroup.h	4670
src/main/decaf/lang/Throwable.h	4670
src/main/decaf/lang/exceptions/ClassCastException.h	4657
src/main/decaf/lang/exceptions/ExceptionDefines.h	4240
src/main/decaf/lang/exceptions/IllegalArgumentException.h	4658
src/main/decaf/lang/exceptions/IllegalMonitorStateException.h	4658



src/main/decaf/lang/exceptions/ <b>IllegalStateException.h</b>	4577
src/main/decaf/lang/exceptions/ <b>IllegalThreadStateException.h</b>	4659
src/main/decaf/lang/exceptions/ <b>IndexOutOfBoundsException.h</b>	4659
src/main/decaf/lang/exceptions/ <b>InterruptedException.h</b>	4659
src/main/decaf/lang/exceptions/ <b>InvalidStateException.h</b>	4660
src/main/decaf/lang/exceptions/ <b>NoSuchElementException.h</b>	4660
src/main/decaf/lang/exceptions/ <b>NullPointerException.h</b>	4661
src/main/decaf/lang/exceptions/ <b>NumberFormatException.h</b>	4661
src/main/decaf/lang/exceptions/ <b>RuntimeException.h</b>	4662
src/main/decaf/lang/exceptions/ <b>UnsupportedOperationException.h</b>	4590
src/main/decaf/net/ <b>BindException.h</b>	4671
src/main/decaf/net/ <b>ConnectException.h</b>	4671
src/main/decaf/net/ <b>HttpRetryException.h</b>	4672
src/main/decaf/net/ <b>Inet4Address.h</b>	4672
src/main/decaf/net/ <b>Inet6Address.h</b>	4672
src/main/decaf/net/ <b>InetAddress.h</b>	4673
src/main/decaf/net/ <b>InetSocketAddress.h</b>	4673
src/main/decaf/net/ <b>MalformedURLException.h</b>	4674
src/main/decaf/net/ <b>NoRouteToHostException.h</b>	4674
src/main/decaf/net/ <b>PortUnreachableException.h</b>	4674
src/main/decaf/net/ <b>ProtocolException.h</b>	4675
src/main/decaf/net/ <b>ServerSocket.h</b>	4675
src/main/decaf/net/ <b>ServerSocketFactory.h</b>	4676
src/main/decaf/net/ <b>Socket.h</b>	4676
src/main/decaf/net/ <b>SocketAddress.h</b>	4677
src/main/decaf/net/ <b>SocketError.h</b>	4678
src/main/decaf/net/ <b>SocketException.h</b>	4678
src/main/decaf/net/ <b>SocketFactory.h</b>	4678
src/main/decaf/net/ <b>SocketImpl.h</b>	4679
src/main/decaf/net/ <b>SocketImplFactory.h</b>	4680
src/main/decaf/net/ <b>SocketOptions.h</b>	4680
src/main/decaf/net/ <b>SocketTimeoutException.h</b>	4680
src/main/decaf/net/ <b>UnknownHostException.h</b>	4684
src/main/decaf/net/ <b>UnknownServiceException.h</b>	4685
src/main/decaf/net/ <b>URI.h</b>	4685
src/main/decaf/net/ <b>URISyntaxException.h</b>	4686
src/main/decaf/net/ <b>URL.h</b>	4686
src/main/decaf/net/ <b>URLDecoder.h</b>	4687
src/main/decaf/net/ <b>URLEncoder.h</b>	4687
src/main/decaf/net/ssl/ <b>SSLContext.h</b>	4681
src/main/decaf/net/ssl/ <b>SSLContextSpi.h</b>	4681
src/main/decaf/net/ssl/ <b>SSLParameters.h</b>	4682
src/main/decaf/net/ssl/ <b>SSLServerSocket.h</b>	4682
src/main/decaf/net/ssl/ <b>SSLServerSocketFactory.h</b>	4683
src/main/decaf/net/ssl/ <b>SSLSocket.h</b>	4683
src/main/decaf/net/ssl/ <b>SSLSocketFactory.h</b>	4684
src/main/decaf/nio/ <b>Buffer.h</b>	4687
src/main/decaf/nio/ <b>BufferOverflowException.h</b>	4688
src/main/decaf/nio/ <b>BufferUnderflowException.h</b>	4688
src/main/decaf/nio/ <b>ByteBuffer.h</b>	4689

src/main/decaf/nio/ <b>CharBuffer.h</b> . . . . .	4689
src/main/decaf/nio/ <b>DoubleBuffer.h</b> . . . . .	4690
src/main/decaf/nio/ <b>FloatBuffer.h</b> . . . . .	4691
src/main/decaf/nio/ <b>IntBuffer.h</b> . . . . .	4691
src/main/decaf/nio/ <b>InvalidMarkException.h</b> . . . . .	4692
src/main/decaf/nio/ <b>LongBuffer.h</b> . . . . .	4692
src/main/decaf/nio/ <b>ReadOnlyBufferException.h</b> . . . . .	4693
src/main/decaf/nio/ <b>ShortBuffer.h</b> . . . . .	4693
src/main/decaf/security/ <b>GeneralSecurityException.h</b> . . . . .	4698
src/main/decaf/security/ <b>InvalidKeyException.h</b> . . . . .	4698
src/main/decaf/security/ <b>Key.h</b> . . . . .	4699
src/main/decaf/security/ <b>KeyException.h</b> . . . . .	4699
src/main/decaf/security/ <b>KeyManagementException.h</b> . . . . .	4700
src/main/decaf/security/ <b>NoSuchAlgorithmException.h</b> . . . . .	4700
src/main/decaf/security/ <b>NoSuchProviderException.h</b> . . . . .	4701
src/main/decaf/security/ <b>Principal.h</b> . . . . .	4701
src/main/decaf/security/ <b>PublicKey.h</b> . . . . .	4701
src/main/decaf/security/ <b>SecureRandom.h</b> . . . . .	4702
src/main/decaf/security/ <b>SecureRandomSpi.h</b> . . . . .	4702
src/main/decaf/security/ <b>SignatureException.h</b> . . . . .	4703
src/main/decaf/security/auth/x500/ <b>X500Principal.h</b> . . . . .	4694
src/main/decaf/security/cert/ <b>Certificate.h</b> . . . . .	4694
src/main/decaf/security/cert/ <b>CertificateEncodingException.h</b> . . . . .	4695
src/main/decaf/security/cert/ <b>CertificateException.h</b> . . . . .	4696
src/main/decaf/security/cert/ <b>CertificateExpiredException.h</b> . . . . .	4696
src/main/decaf/security/cert/ <b>CertificateNotYetValidException.h</b> . . . . .	4697
src/main/decaf/security/cert/ <b>CertificateParsingException.h</b> . . . . .	4697
src/main/decaf/security/cert/ <b>X509Certificate.h</b> . . . . .	4698
src/main/decaf/util/ <b>AbstractCollection.h</b> . . . . .	4703
src/main/decaf/util/ <b>AbstractList.h</b> . . . . .	4704
src/main/decaf/util/ <b>AbstractMap.h</b> . . . . .	4705
src/main/decaf/util/ <b>AbstractQueue.h</b> . . . . .	4705
src/main/decaf/util/ <b>AbstractSequentialList.h</b> . . . . .	4706
src/main/decaf/util/ <b>AbstractSet.h</b> . . . . .	4707
src/main/decaf/util/ <b>Collection.h</b> . . . . .	4707
src/main/decaf/util/ <b>Comparator.h</b> . . . . .	4708
src/main/decaf/util/ <b>Config.h</b> . . . . .	4275
src/main/decaf/util/ <b>Date.h</b> . . . . .	4729
src/main/decaf/util/ <b>Iterator.h</b> . . . . .	4729
src/main/decaf/util/ <b>List.h</b> . . . . .	4730
src/main/decaf/util/ <b>ListIterator.h</b> . . . . .	4730
src/main/decaf/util/ <b>Map.h</b> . . . . .	4742
src/main/decaf/util/ <b>PriorityQueue.h</b> . . . . .	4743
src/main/decaf/util/ <b>Properties.h</b> . . . . .	4743
src/main/decaf/util/ <b>Queue.h</b> . . . . .	4585
src/main/decaf/util/ <b>Random.h</b> . . . . .	4744
src/main/decaf/util/ <b>Set.h</b> . . . . .	4745
src/main/decaf/util/ <b>StlList.h</b> . . . . .	4745
src/main/decaf/util/ <b>StlMap.h</b> . . . . .	4746
src/main/decaf/util/ <b>StlQueue.h</b> . . . . .	4747

src/main/decaf/util/ <b>StdSet.h</b>	4747
src/main/decaf/util/ <b>StringTokenizer.h</b>	4748
src/main/decaf/util/ <b>Timer.h</b>	4748
src/main/decaf/util/ <b>TimerTask.h</b>	4749
src/main/decaf/util/ <b>UUID.h</b>	4750
src/main/decaf/util/comparators/ <b>Less.h</b>	4708
src/main/decaf/util/concurrent/ <b>BlockingQueue.h</b>	4711
src/main/decaf/util/concurrent/ <b>BrokenBarrierException.h</b>	4712
src/main/decaf/util/concurrent/ <b>Callable.h</b>	4712
src/main/decaf/util/concurrent/ <b>CancellationException.h</b>	4712
src/main/decaf/util/concurrent/ <b>Concurrent.h</b>	4713
src/main/decaf/util/concurrent/ <b>ConcurrentMap.h</b>	4714
src/main/decaf/util/concurrent/ <b>ConcurrentStlMap.h</b>	4715
src/main/decaf/util/concurrent/ <b>CountDownLatch.h</b>	4715
src/main/decaf/util/concurrent/ <b>Delayed.h</b>	4716
src/main/decaf/util/concurrent/ <b>ExecutionException.h</b>	4716
src/main/decaf/util/concurrent/ <b>Executor.h</b>	4717
src/main/decaf/util/concurrent/ <b>ExecutorService.h</b>	4717
src/main/decaf/util/concurrent/ <b>Future.h</b>	4718
src/main/decaf/util/concurrent/ <b>Lock.h</b>	4718
src/main/decaf/util/concurrent/ <b>Mutex.h</b>	4722
src/main/decaf/util/concurrent/ <b>PooledThread.h</b>	4722
src/main/decaf/util/concurrent/ <b>PooledThreadListener.h</b>	4723
src/main/decaf/util/concurrent/ <b>RejectedExecutionException.h</b>	4723
src/main/decaf/util/concurrent/ <b>RejectedExecutionHandler.h</b>	4724
src/main/decaf/util/concurrent/ <b>Semaphore.h</b>	4724
src/main/decaf/util/concurrent/ <b>Synchronizable.h</b>	4725
src/main/decaf/util/concurrent/ <b>SynchronousQueue.h</b>	4725
src/main/decaf/util/concurrent/ <b>TaskListener.h</b>	4726
src/main/decaf/util/concurrent/ <b>ThreadFactory.h</b>	4726
src/main/decaf/util/concurrent/ <b>ThreadPool.h</b>	4727
src/main/decaf/util/concurrent/ <b>TimeoutException.h</b>	4728
src/main/decaf/util/concurrent/ <b>TimeUnit.h</b>	4728
src/main/decaf/util/concurrent/atomic/ <b>AtomicBoolean.h</b>	4709
src/main/decaf/util/concurrent/atomic/ <b>AtomicInteger.h</b>	4709
src/main/decaf/util/concurrent/atomic/ <b>AtomicRefCounter.h</b>	4710
src/main/decaf/util/concurrent/atomic/ <b>AtomicReference.h</b>	4710
src/main/decaf/util/concurrent/locks/ <b>Condition.h</b>	4719
src/main/decaf/util/concurrent/locks/ <b>Lock.h</b>	4719
src/main/decaf/util/concurrent/locks/ <b>LockSupport.h</b>	4720
src/main/decaf/util/concurrent/locks/ <b>ReadWriteLock.h</b>	4721
src/main/decaf/util/concurrent/locks/ <b>ReentrantLock.h</b>	4721
src/main/decaf/util/logging/ <b>ConsoleHandler.h</b>	4731
src/main/decaf/util/logging/ <b>ErrorManager.h</b>	4731
src/main/decaf/util/logging/ <b>Filter.h</b>	4732
src/main/decaf/util/logging/ <b>Formatter.h</b>	4732
src/main/decaf/util/logging/ <b>Handler.h</b>	4733
src/main/decaf/util/logging/ <b>Level.h</b>	4734
src/main/decaf/util/logging/ <b>Logger.h</b>	4734
src/main/decaf/util/logging/ <b>LoggerCommon.h</b>	4735

src/main/decaf/util/logging/ <b>LoggerDefines.h</b> . . . . .	4735
src/main/decaf/util/logging/ <b>LoggerHierarchy.h</b> . . . . .	4737
src/main/decaf/util/logging/ <b>LogManager.h</b> . . . . .	4737
src/main/decaf/util/logging/ <b>LogRecord.h</b> . . . . .	4738
src/main/decaf/util/logging/ <b>LogWriter.h</b> . . . . .	4739
src/main/decaf/util/logging/ <b>MarkBlockLogger.h</b> . . . . .	4739
src/main/decaf/util/logging/ <b>PropertiesChangeListener.h</b> . . . . .	4740
src/main/decaf/util/logging/ <b>SimpleFormatter.h</b> . . . . .	4740
src/main/decaf/util/logging/ <b>SimpleLogger.h</b> . . . . .	4741
src/main/decaf/util/logging/ <b>StreamHandler.h</b> . . . . .	4741
src/main/decaf/util/logging/ <b>XMLFormatter.h</b> . . . . .	4742
src/main/decaf/util/zip/ <b>Adler32.h</b> . . . . .	4750
src/main/decaf/util/zip/ <b>CheckedInputStream.h</b> . . . . .	4751
src/main/decaf/util/zip/ <b>CheckedOutputStream.h</b> . . . . .	4751
src/main/decaf/util/zip/ <b>Checksum.h</b> . . . . .	4752
src/main/decaf/util/zip/ <b>CRC32.h</b> . . . . .	4752
src/main/decaf/util/zip/ <b>DataFormatException.h</b> . . . . .	4753
src/main/decaf/util/zip/ <b>Deflater.h</b> . . . . .	4753
src/main/decaf/util/zip/ <b>DeflaterOutputStream.h</b> . . . . .	4754
src/main/decaf/util/zip/ <b>Inflater.h</b> . . . . .	4754
src/main/decaf/util/zip/ <b>InflaterInputStream.h</b> . . . . .	4755
src/main/decaf/util/zip/ <b>ZipException.h</b> . . . . .	4756

## Chapter 5

# Namespace Documentation

### 5.1 activemq Namespace Reference

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

#### Namespaces

- namespace **cmsutil**
- namespace **commands**
- namespace **core**
- namespace **exceptions**
- namespace **io**
- namespace **library**
- namespace **state**
- namespace **threads**
- namespace **transport**
- namespace **util**
- namespace **wireformat**

#### 5.1.1 Detailed Description

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CON-

DITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## 5.2 activemq::cmsutil Namespace Reference

### Data Structures

- class **CachedConsumer**  
*A cached message consumer contained within a pooled session.*
- class **CachedProducer**  
*A cached message producer contained within a pooled session.*
- class **CmsAccessor**  
*Base class for **activemq.cmsutil.CmsTemplate** (p. 1201) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 1361) to operate on.*
- class **CmsDestinationAccessor**  
*Extends the **CmsAccessor** (p. 1183) to add support for resolving destination names.*
- class **CmsTemplate**  
***CmsTemplate** (p. 1201) simplifies performing synchronous CMS operations.*
- class **DestinationResolver**  
*Resolves a CMS destination name to a *Destination*.*
- class **DynamicDestinationResolver**  
*Resolves a CMS destination name to a *Destination*.*
- class **MessageCreator**  
*Creates the user-defined message to be sent by the **CmsTemplate** (p. 1201).*
- class **PooledSession**  
*A pooled session object that wraps around a delegate session.*
- class **ProducerCallback**  
*Callback for sending a message to a CMS destination.*
- class **ResourceLifecycleManager**  
*Manages the lifecycle of a set of CMS resources.*
- class **SessionCallback**  
*Callback for executing any number of operations on a provided CMS Session.*

- class **SessionPool**

*A pool of CMS sessions from the same connection and with the same acknowledge mode.*

## 5.3 activemq::commands Namespace Reference

### Data Structures

- class **ActiveMQBlobMessage**
- class **ActiveMQBytesMessage**
- class **ActiveMQDestination**
- class **ActiveMQMapMessage**
- class **ActiveMQMessage**
- class **ActiveMQMessageTemplate**
- class **ActiveMQObjectMessage**
- class **ActiveMQQueue**
- class **ActiveMQStreamMessage**
- class **ActiveMQTempDestination**
- class **ActiveMQTempQueue**
- class **ActiveMQTempTopic**
- class **ActiveMQTextMessage**
- class **ActiveMQTopic**
- class **BaseCommand**
- class **BaseDataStructure**
- class **BooleanExpression**
- class **BrokerError**

*This class represents an Exception sent from the Broker.*

- class **BrokerId**
- class **BrokerInfo**
- class **Command**
- class **ConnectionControl**
- class **ConnectionError**
- class **ConnectionId**
- class **ConnectionInfo**
- class **ConsumerControl**
- class **ConsumerId**
- class **ConsumerInfo**
- class **ControlCommand**
- class **DataArrayResponse**
- class **DataResponse**
- class **DataStructure**
- class **DestinationInfo**
- class **DiscoveryEvent**

- class **ExceptionResponse**
- class **FlushCommand**
- class **IntegerResponse**
- class **JournalQueueAck**
- class **JournalTopicAck**
- class **JournalTrace**
- class **JournalTransaction**
- class **KeepAliveInfo**
- class **LastPartialCommand**
- class **LocalTransactionId**
- class **Message**
- class **MessageAck**
- class **MessageDispatch**
- class **MessageDispatchNotification**
- class **MessageId**
- class **MessagePull**
- class **NetworkBridgeFilter**
- class **PartialCommand**
- class **ProducerAck**
- class **ProducerId**
- class **ProducerInfo**
- class **RemoveInfo**
- class **RemoveSubscriptionInfo**
- class **ReplayCommand**
- class **Response**
- class **SessionId**
- class **SessionInfo**
- class **ShutdownInfo**
- class **SubscriptionInfo**
- class **TransactionId**
- class **TransactionInfo**
- class **WireFormatInfo**
- class **XATransactionId**

## 5.4 activemq::core Namespace Reference

### Namespaces

- namespace **policies**



## Data Structures

- class **ActiveMQAckHandler**  
*Interface class that is used to give CMS Messages an interface to Ack themselves with.*
- class **ActiveMQConnection**  
*Concrete connection used for all connectors to the ActiveMQ broker.*
- class **ActiveMQConnectionFactory**
- class **ActiveMQConnectionMetaData**  
*This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 260) class.*
- class **ActiveMQConstants**  
*Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values.*
- class **ActiveMQConsumer**
- class **ActiveMQProducer**
- class **ActiveMQQueueBrowser**
- class **ActiveMQSession**
- class **ActiveMQSessionExecutor**  
*Delegate dispatcher for a single session.*
- class **ActiveMQTransactionContext**  
*Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.*
- class **DispatchData**  
*Simple POCO that contains the information necessary to route a message to a specified consumer.*
- class **Dispatcher**  
*Interface for an object responsible for dispatching messages to consumers.*
- class **MessageDispatchChannel**
- class **PrefetchPolicy**  
*Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.*
- class **RedeliveryPolicy**  
*Interface for a **RedeliveryPolicy** (p. 3267) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.*
- class **Synchronization**  
*Transacted Object **Synchronization** (p. 3826), used to sync the events of a Transaction with the items in the Transaction.*

## 5.5 activemq::core::policies Namespace Reference

### Data Structures

- class **DefaultPrefetchPolicy**
- class **DefaultRedeliveryPolicy**

## 5.6 activemq::exceptions Namespace Reference

### Data Structures

- class **ActiveMQException**
- class **BrokerException**

## 5.7 activemq::io Namespace Reference

### Data Structures

- class **LoggingInputStream**
- class **LoggingOutputStream**  
*OutputStream filter that just logs the data being written.*

## 5.8 activemq::library Namespace Reference

### Data Structures

- class **ActiveMQCPP**

## 5.9 activemq::state Namespace Reference

### Data Structures

- class **CommandVisitor**  
*Interface for an Object that can visit the various Command Objects that are sent from and to this client.*
- class **CommandVisitorAdapter**  
*Default Implementation of a **CommandVisitor** (p.1233) that returns NULL for all calls.*
- class **ConnectionState**

- class **ConnectionStateTracker**
- class **ConsumerState**
- class **ProducerState**
- class **SessionState**
- class **Tracked**
- class **TransactionState**

## 5.10 activemq::threads Namespace Reference

### Data Structures

- class **CompositeTask**  
*Represents a single task that can be part of a set of Tasks that are contained in a **CompositeTaskRunner** (p. 1256).*
- class **CompositeTaskRunner**  
*A **Task** (p. 3846) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.*
- class **DedicatedTaskRunner**
- class **Task**  
*Represents a unit of work that requires one or more iterations to complete.*
- class **TaskRunner**

## 5.11 activemq::transport Namespace Reference

### Namespaces

- namespace **correlator**
- namespace **failover**
- namespace **inactivity**
- namespace **logging**
- namespace **mock**
- namespace **tcp**

### Data Structures

- class **AbstractTransportFactory**  
*Abstract implementation of the **TransportFactory** (p. 4003) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 4003) instances.*
- class **CompositeTransport**

A Composite **Transport** (p. 3996) is a **Transport** (p. 3996) implementation that is composed of several **Transports**.

- class **DefaultTransportListener**
- class **IOTransport**

Implementation of the **Transport** (p. 3996) interface that performs marshaling of commands to IO streams.

- class **Transport**

Interface for a transport layer for command objects.

- class **TransportFactory**

Defines the interface for Factories that create **Transports** or **TransportFilters**.

- class **TransportFilter**

A filter on the transport layer.

- class **TransportListener**

A listener of asynchronous exceptions from a command transport object.

- class **TransportRegistry**

Registry of all **Transport** (p. 3996) Factories that are available to the client at runtime.

## 5.12 activemq::transport::correlator Namespace Reference

### Data Structures

- class **FutureResponse**

A container that holds a response object.

- class **ResponseCorrelator**

This type of transport filter is responsible for correlating asynchronous responses with requests.

## 5.13 activemq::transport::failover Namespace Reference

### Data Structures

- class **BackupTransport**
- class **BackupTransportPool**
- class **CloseTransportsTask**
- class **FailoverTransport**

- class **FailoverTransportFactory**  
*Creates an instance of a **FailoverTransport** (p. 1930).*
- class **FailoverTransportListener**  
*Utility class used by the **Transport** (p. 3996) to perform the work of responding to events from the active **Transport** (p. 3996).*
- class **URIPool**

## 5.14 activemq::transport::inactivity Namespace Reference

### Data Structures

- class **InactivityMonitor**
- class **ReadChecker**  
*Runnable class that is used by the {.*
- class **WriteChecker**  
*Runnable class used by the {.*

## 5.15 activemq::transport::logging Namespace Reference

### Data Structures

- class **LoggingTransport**  
*A transport filter that logs commands as they are sent/received.*

## 5.16 activemq::transport::mock Namespace Reference

### Data Structures

- class **InternalCommandListener**  
*Listens for Commands sent from the **MockTransport** (p. 2854).*
- class **MockTransport**  
*The **MockTransport** (p. 2854) defines a base level **Transport** (p. 3996) class that is intended to be used in place of an a regular protocol **Transport** (p. 3996) such as TCP.*
- class **MockTransportFactory**

*Manufactures MockTransports, which are objects that read from input streams and write to output streams.*

- class **ResponseBuilder**

*Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.*

## 5.17 activemq::transport::tcp Namespace Reference

### Data Structures

- class **SslTransport**

*Transport (p. 3996) for connecting to a Broker using an SSL Socket.*

- class **SslTransportFactory**

- class **TcpTransport**

*Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 2213).*

- class **TcpTransportFactory**

*Factory Responsible for creating the **TcpTransport** (p. 3865).*

## 5.18 activemq::util Namespace Reference

### Data Structures

- class **ActiveMQProperties**

*Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 3216) object.*

- class **CMSExceptionSupport**

- class **CompositeData**

*Represents a Composite URI.*

- class **IdGenerator**

- class **LongSequenceGenerator**

*This class is used to generate a sequence of long long values that are incremented each time a new value is requested.*

- class **MarshallingSupport**

- class **MemoryUsage**

- class **PrimitiveList**

*List of primitives.*

- class **PrimitiveMap**  
*Map of named primitives.*
- class **PrimitiveValueConverter**  
*Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 3099) from one type to another.*
- class **PrimitiveValueNode**  
*Class that wraps around a single value of one of the many types.*
- class **URISupport**
- class **Usage**

## 5.19 activemq::wireformat Namespace Reference

### Namespaces

- namespace **openwire**
- namespace **stomp**

### Data Structures

- class **MarshalAware**
- class **WireFormat**  
*Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.*
- class **WireFormatFactory**  
*The **WireFormatFactory** (p. 4092) is the interface that all **WireFormatFactory** (p. 4092) classes must extend.*
- class **WireFormatNegotiator**  
*Defines a **WireFormatNegotiator** (p. 4129) which allows a **WireFormat** (p. 4088) to.*
- class **WireFormatRegistry**  
*Registry of all **WireFormat** (p. 4088) Factories that are available to the client at run-time.*

## 5.20 activemq::wireformat::openwire Namespace Reference

### Namespaces

- namespace **marshal**

- namespace **utils**

### Data Structures

- class **OpenWireFormat**
- class **OpenWireFormatFactory**
- class **OpenWireFormatNegotiator**
- class **OpenWireResponseBuilder**

## 5.21 activemq::wireformat::openwire::marshal Namespace Reference

### Namespaces

- namespace **v1**
- namespace **v2**
- namespace **v3**
- namespace **v4**
- namespace **v5**
- namespace **v6**

### Data Structures

- class **BaseDataStreamMarshaller**  
*Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol.*
- class **DataStreamMarshaller**  
*Base class for all classes that marshal commands for Openwire.*
- class **PrimitiveTypesMarshaller**  
*This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.*

## 5.22 activemq::wireformat::openwire::marshal::v1 Namespace Reference

### Data Structures

- class **ActiveMQBlobMessageMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 194).*
- class **ActiveMQBytesMessageMarshaller**



*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 239).*

- class **ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 328).*

- class **ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 369).*

- class **ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 397).*

- class **ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 445).*

- class **ActiveMQQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 491).*

- class **ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 557).*

- class **ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 587).*

- class **ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 615).*

- class **ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 649).*

- class **ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 680).*

- class **ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 710).*

- class **BaseCommandMarshaller**

*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 786).*

- class **BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 886).*

- class **BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 919).*

- class **ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1315).*

- class **ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1349).*

- class **ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1381).*

- class **ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1412).*

- class **ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1459).*

- class **ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1489).*

- class **ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1522).*

- class **ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1552).*

- class **DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1587).*

- class **DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1656).*

- class **DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1797).*

- class **DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1831).*

- class **ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1919).*

- class **FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2019).*

- class **IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2180).*

- class **JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2248).*

- class **JournalTopicAckMarshaller**  
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2277).*
- class **JournalTraceMarshaller**  
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2300).*
- class **JournalTransactionMarshaller**  
*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2331).*
- class **KeepAliveInfoMarshaller**  
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2359).*
- class **LastPartialCommandMarshaller**  
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2395).*
- class **LocalTransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2446).*
- class **MarshallerFactory**  
*Used to create marshallers for a specific version of the wire protocol.*
- class **MessageAckMarshaller**  
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2665).*
- class **MessageDispatchMarshaller**  
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2707).*
- class **MessageDispatchNotificationMarshaller**  
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2737).*
- class **MessageIdMarshaller**  
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2775).*
- class **MessageMarshaller**  
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2798).*
- class **MessagePullMarshaller**  
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2845).*
- class **NetworkBridgeFilterMarshaller**  
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2901).*
- class **PartialCommandMarshaller**  
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3028).*

- class **ProducerAckMarshaller**  
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3150).*
- class **ProducerIdMarshaller**  
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3181).*
- class **ProducerInfoMarshaller**  
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3199).*
- class **RemoveInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3300).*
- class **RemoveSubscriptionInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3318).*
- class **ReplayCommandMarshaller**  
*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3351).*
- class **ResponseMarshaller**  
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3408).*
- class **SessionIdMarshaller**  
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3501).*
- class **SessionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3517).*
- class **ShutdownInfoMarshaller**  
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3584).*
- class **SubscriptionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3791).*
- class **TransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3939).*
- class **TransactionInfoMarshaller**  
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3968).*
- class **WireFormatInfoMarshaller**  
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 4121).*
- class **XATransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 4160).*

## 5.23 activemq::wireformat::openwire::marshal::v2 Namespace Reference

### Data Structures

- class **ActiveMQBlobMessageMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 202).*
- class **ActiveMQBytesMessageMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 256).*
- class **ActiveMQDestinationMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 340).*
- class **ActiveMQMapMessageMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 381).*
- class **ActiveMQMessageMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 410).*
- class **ActiveMQObjectMessageMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 457).*
- class **ActiveMQQueueMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 504).*
- class **ActiveMQStreamMessageMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQStreamMessageMarshaller (p. 570).*
- class **ActiveMQTempDestinationMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQTempDestinationMarshaller (p. 599).*
- class **ActiveMQTempQueueMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQTempQueueMarshaller (p. 628).*
- class **ActiveMQTempTopicMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQTempTopicMarshaller (p. 658).*
- class **ActiveMQTextMessageMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 693).*
- class **ActiveMQTopicMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQTopicMarshaller (p. 722).*

- class **BaseCommandMarshaller**  
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 807).*
- class **BrokerIdMarshaller**  
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 898).*
- class **BrokerInfoMarshaller**  
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 932).*
- class **ConnectionControlMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1328).*
- class **ConnectionErrorMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1336).*
- class **ConnectionIdMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1368).*
- class **ConnectionInfoMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1400).*
- class **ConsumerControlMarshaller**  
*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1447).*
- class **ConsumerIdMarshaller**  
*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1477).*
- class **ConsumerInfoMarshaller**  
*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1510).*
- class **ControlCommandMarshaller**  
*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1539).*
- class **DataArrayResponseMarshaller**  
*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1574).*
- class **DataResponseMarshaller**  
*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1643).*
- class **DestinationInfoMarshaller**  
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1784).*
- class **DiscoveryEventMarshaller**  
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1819).*
- class **ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1902).*

- class **FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2006).*

- class **IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2167).*

- class **JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2232).*

- class **JournalTopicAckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2261).*

- class **JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2283).*

- class **JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2315).*

- class **KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2342).*

- class **LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2382).*

- class **LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2429).*

- class **MarshallerFactory**

*Used to create marshallers for a specific version of the wire protocol.*

- class **MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2653).*

- class **MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2690).*

- class **MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2725).*

- class **MessageIdMarshaller**

*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2755).*

- class **MessageMarshaller**

*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2789).*

- class **MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2828).*

- class **NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2881).*

- class **PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3010).*

- class **ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3128).*

- class **ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3161).*

- class **ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3194).*

- class **RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3288).*

- class **RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3326).*

- class **ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3355).*

- class **ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3393).*

- class **SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3481).*

- class **SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3525).*

- class **ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3580).*

- class **SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3807).*

- class **TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3943).*



- class **TransactionInfoMarshaller**  
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3985).*
- class **WireFormatInfoMarshaller**  
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 4113).*
- class **XATransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 4151).*

## 5.24 activemq::wireformat::openwire::marshal::v3 Namespace Reference

### Data Structures

- class **ActiveMQBlobMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 189).*
- class **ActiveMQBytesMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 235).*
- class **ActiveMQDestinationMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 324).*
- class **ActiveMQMapMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 364).*
- class **ActiveMQMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 393).*
- class **ActiveMQObjectMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 440).*
- class **ActiveMQQueueMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 487).*
- class **ActiveMQStreamMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 553).*
- class **ActiveMQTempDestinationMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 582).*

- class **ActiveMQTempQueueMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 611).*
- class **ActiveMQTempTopicMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 641).*
- class **ActiveMQTextMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 671).*
- class **ActiveMQTopicMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 701).*
- class **BaseCommandMarshaller**  
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 771).*
- class **BrokerIdMarshaller**  
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 878).*
- class **BrokerInfoMarshaller**  
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 910).*
- class **ConnectionControlMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1307).*
- class **ConnectionErrorMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1340).*
- class **ConnectionIdMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1373).*
- class **ConnectionInfoMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1404).*
- class **ConsumerControlMarshaller**  
*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1451).*
- class **ConsumerIdMarshaller**  
*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1481).*
- class **ConsumerInfoMarshaller**  
*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1514).*
- class **ControlCommandMarshaller**  
*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1544).*
- class **DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1579).*

- class **DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1647).*

- class **DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1789).*

- class **DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1823).*

- class **ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1906).*

- class **FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2010).*

- class **IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2171).*

- class **JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2240).*

- class **JournalTopicAckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2265).*

- class **JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2288).*

- class **JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2319).*

- class **KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2347).*

- class **LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2378).*

- class **LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2433).*

- class **MarshallerFactory**

*Used to create marshallers for a specific version of the wire protocol.*

- class **MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2657).*

- class **MessageDispatchMarshaller**  
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2695).*
- class **MessageDispatchNotificationMarshaller**  
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2729).*
- class **MessageIdMarshaller**  
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2767).*
- class **MessageMarshaller**  
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2785).*
- class **MessagePullMarshaller**  
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2837).*
- class **NetworkBridgeFilterMarshaller**  
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2893).*
- class **PartialCommandMarshaller**  
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3019).*
- class **ProducerAckMarshaller**  
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3137).*
- class **ProducerIdMarshaller**  
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3169).*
- class **ProducerInfoMarshaller**  
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3207).*
- class **RemoveInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3296).*
- class **RemoveSubscriptionInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3322).*
- class **ReplayCommandMarshaller**  
*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3360).*
- class **ResponseMarshaller**  
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3403).*
- class **SessionIdMarshaller**  
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3497).*

- class **SessionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3521).*
- class **ShutdownInfoMarshaller**  
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3593).*
- class **SubscriptionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3786).*
- class **TransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3947).*
- class **TransactionInfoMarshaller**  
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3972).*
- class **WireFormatInfoMarshaller**  
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 4125).*
- class **XATransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 4164).*

## 5.25 activemq::wireformat::openwire::marshal::v4 Namespace Reference

### Data Structures

- class **ActiveMQBlobMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 198).*
- class **ActiveMQBytesMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 243).*
- class **ActiveMQDestinationMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 332).*
- class **ActiveMQMapMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 373).*
- class **ActiveMQMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 401).*
- class **ActiveMQObjectMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 449).*

- class **ActiveMQQueueMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 495).*
- class **ActiveMQStreamMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 561).*
- class **ActiveMQTempDestinationMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 591).*
- class **ActiveMQTempQueueMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 619).*
- class **ActiveMQTempTopicMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 645).*
- class **ActiveMQTextMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 676).*
- class **ActiveMQTopicMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 705).*
- class **BaseCommandMarshaller**  
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 778).*
- class **BrokerIdMarshaller**  
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 882).*
- class **BrokerInfoMarshaller**  
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 915).*
- class **ConnectionControlMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1311).*
- class **ConnectionErrorMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1344).*
- class **ConnectionIdMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1377).*
- class **ConnectionInfoMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1408).*
- class **ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1455).*

- class **ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1485).*

- class **ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1518).*

- class **ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1548).*

- class **DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1583).*

- class **DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1652).*

- class **DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1793).*

- class **DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1827).*

- class **ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1915).*

- class **FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2014).*

- class **IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2175).*

- class **JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2244).*

- class **JournalTopicAckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2273).*

- class **JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2296).*

- class **JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2327).*

- class **KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2351).*

- class **LastPartialCommandMarshaller**  
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2391).*
- class **LocalTransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2442).*
- class **MarshallerFactory**  
*Used to create marshallers for a specific version of the wire protocol.*
- class **MessageAckMarshaller**  
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2661).*
- class **MessageDispatchMarshaller**  
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2703).*
- class **MessageDispatchNotificationMarshaller**  
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2733).*
- class **MessageIdMarshaller**  
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2759).*
- class **MessageMarshaller**  
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2793).*
- class **MessagePullMarshaller**  
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2841).*
- class **NetworkBridgeFilterMarshaller**  
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2897).*
- class **PartialCommandMarshaller**  
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3023).*
- class **ProducerAckMarshaller**  
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3133).*
- class **ProducerIdMarshaller**  
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3165).*
- class **ProducerInfoMarshaller**  
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3190).*
- class **RemoveInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3309).*



- class **RemoveSubscriptionInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3339).*
- class **ReplayCommandMarshaller**  
*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3347).*
- class **ResponseMarshaller**  
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3388).*
- class **SessionIdMarshaller**  
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3485).*
- class **SessionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3530).*
- class **ShutdownInfoMarshaller**  
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3597).*
- class **SubscriptionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3799).*
- class **TransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3951).*
- class **TransactionInfoMarshaller**  
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3981).*
- class **WireFormatInfoMarshaller**  
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 4117).*
- class **XATransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 4155).*

## 5.26 activemq::wireformat::openwire::marshal::v5 Namespace Reference

### Data Structures

- class **ActiveMQBlobMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 206).*
- class **ActiveMQBytesMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 247).*

- class **ActiveMQDestinationMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 336).*
- class **ActiveMQMapMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 377).*
- class **ActiveMQMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 405).*
- class **ActiveMQObjectMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 453).*
- class **ActiveMQQueueMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 499).*
- class **ActiveMQStreamMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 566).*
- class **ActiveMQTempDestinationMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 595).*
- class **ActiveMQTempQueueMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 624).*
- class **ActiveMQTempTopicMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 654).*
- class **ActiveMQTextMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 684).*
- class **ActiveMQTopicMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 714).*
- class **BaseCommandMarshaller**  
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 793).*
- class **BrokerIdMarshaller**  
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 890).*
- class **BrokerInfoMarshaller**  
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 923).*
- class **ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1319).*

- class **ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1353).*

- class **ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1385).*

- class **ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1417).*

- class **ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1464).*

- class **ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1493).*

- class **ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1527).*

- class **ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1556).*

- class **DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1592).*

- class **DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1635).*

- class **DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1806).*

- class **DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1835).*

- class **ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1910).*

- class **FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2023).*

- class **IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2184).*

- class **JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2236).*

- class **JournalTopicAckMarshaller**  
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2256).*
- class **JournalTraceMarshaller**  
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2304).*
- class **JournalTransactionMarshaller**  
*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2323).*
- class **KeepAliveInfoMarshaller**  
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2355).*
- class **LastPartialCommandMarshaller**  
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2386).*
- class **LocalTransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2437).*
- class **MarshallerFactory**  
*Used to create marshallers for a specific version of the wire protocol.*
- class **MessageAckMarshaller**  
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2670).*
- class **MessageDispatchMarshaller**  
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2699).*
- class **MessageDispatchNotificationMarshaller**  
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2742).*
- class **MessageIdMarshaller**  
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2763).*
- class **MessageMarshaller**  
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2780).*
- class **MessagePullMarshaller**  
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2833).*
- class **NetworkBridgeFilterMarshaller**  
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2889).*
- class **PartialCommandMarshaller**  
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3014).*

- class **ProducerAckMarshaller**  
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3141).*
- class **ProducerIdMarshaller**  
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3173).*
- class **ProducerInfoMarshaller**  
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3203).*
- class **RemoveInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3305).*
- class **RemoveSubscriptionInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3335).*
- class **ReplayCommandMarshaller**  
*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3368).*
- class **ResponseMarshaller**  
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3398).*
- class **SessionIdMarshaller**  
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3493).*
- class **SessionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3513).*
- class **ShutdownInfoMarshaller**  
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3589).*
- class **SubscriptionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3795).*
- class **TransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3935).*
- class **TransactionInfoMarshaller**  
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3964).*
- class **WireFormatInfoMarshaller**  
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 4104).*
- class **XATransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 4168).*

## 5.27 activemq::wireformat::openwire::marshal::v6 Namespace Reference

### Data Structures

- class **ActiveMQBlobMessageMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller* (p. 211).
- class **ActiveMQBytesMessageMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller* (p. 252).
- class **ActiveMQDestinationMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller* (p. 344).
- class **ActiveMQMapMessageMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller* (p. 386).
- class **ActiveMQMessageMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQMessageMarshaller* (p. 414).
- class **ActiveMQObjectMessageMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller* (p. 462).
- class **ActiveMQQueueMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQQueueMarshaller* (p. 508).
- class **ActiveMQStreamMessageMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQStreamMessageMarshaller* (p. 574).
- class **ActiveMQTempDestinationMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQTempDestinationMarshaller* (p. 603).
- class **ActiveMQTempQueueMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQTempQueueMarshaller* (p. 632).
- class **ActiveMQTempTopicMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQTempTopicMarshaller* (p. 662).
- class **ActiveMQTextMessageMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller* (p. 688).
- class **ActiveMQTopicMarshaller**  
*Marshaling code for Open Wire Format for ActiveMQTopicMarshaller* (p. 718).

- class **BaseCommandMarshaller**  
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 800).*
- class **BrokerIdMarshaller**  
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 894).*
- class **BrokerInfoMarshaller**  
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 927).*
- class **ConnectionControlMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1324).*
- class **ConnectionErrorMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1357).*
- class **ConnectionIdMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1389).*
- class **ConnectionInfoMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1421).*
- class **ConsumerControlMarshaller**  
*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1468).*
- class **ConsumerIdMarshaller**  
*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1497).*
- class **ConsumerInfoMarshaller**  
*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1531).*
- class **ControlCommandMarshaller**  
*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1561).*
- class **DataArrayResponseMarshaller**  
*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1596).*
- class **DataResponseMarshaller**  
*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1639).*
- class **DestinationInfoMarshaller**  
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1801).*
- class **DiscoveryEventMarshaller**  
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1815).*
- class **ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1898).*

- class **FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2002).*

- class **IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2162).*

- class **JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2228).*

- class **JournalTopicAckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2269).*

- class **JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2292).*

- class **JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2311).*

- class **KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2338).*

- class **LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2374).*

- class **LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2425).*

- class **MarshallerFactory**

*Used to create marshallers for a specific version of the wire protocol.*

- class **MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2648).*

- class **MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2712).*

- class **MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2720).*

- class **MessageIdMarshaller**

*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2771).*

- class **MessageMarshaller**



*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2802).*

- class **MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2850).*

- class **NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2885).*

- class **PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3005).*

- class **ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3145).*

- class **ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3177).*

- class **ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3211).*

- class **RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3292).*

- class **RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3331).*

- class **ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3364).*

- class **ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3413).*

- class **SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3489).*

- class **SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3508).*

- class **ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3576).*

- class **SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3803).*

- class **TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3955).*

- class **TransactionInfoMarshaller**  
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3977).*
- class **WireFormatInfoMarshaller**  
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 4109).*
- class **XATransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 4147).*

## 5.28 activemq::wireformat::openwire::utils Namespace Reference

### Data Structures

- class **BooleanStream**  
*Manages the writing and reading of boolean data streams to and from a data source such as a `DataInputStream` or `DataOutputStream`.*
- class **HexTable**  
*The **HexTable** (p. 2048) class maps hexadecimal strings to the value of an index into the table, i.e.*
- class **MessagePropertyInterceptor**  
*Used the base `ActiveMQMessage` class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the `OpenWire` Message properties.*

## 5.29 activemq::wireformat::stomp Namespace Reference

### Data Structures

- class **StompCommandConstants**
- class **StompFrame**  
*A Stomp-level message frame that encloses all messages to and from the broker.*
- class **StompHelper**  
*Utility Methods used when marshaling to and from `StompFrame`'s.*
- class **StompWireFormat**
- class **StompWireFormatFactory**  
*Factory used to create the Stomp Wire Format instance.*

## 5.30 cms Namespace Reference

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

### Data Structures

- class **BytesMessage**  
*A **BytesMessage** (p. 1079) object is used to send a message containing a stream of unsigned bytes.*
- class **Closeable**  
*Interface for a class that implements the close method.*
- class **CMSException**  
*CMS API Exception that is the base for all exceptions thrown from CMS classes.*
- class **CMSProperties**  
*Interface for a Java-like properties object.*
- class **CMSSecurityException**  
*This exception must be thrown when a provider rejects a user name/password submitted by a client.*
- class **Connection**  
*The client's connection to its provider.*
- class **ConnectionFactory**  
*Defines the interface for a factory that creates connection objects, the **Connection** (p. 1296) objects returned implement the CMS **Connection** (p. 1296) interface and hide the CMS Provider specific implementation details behind that interface.*
- class **ConnectionMetaData**  
*A **ConnectionMetaData** (p. 1425) object provides information describing the **Connection** (p. 1296) object.*
- class **DeliveryMode**  
*This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.*
- class **Destination**  
*A **Destination** (p. 1776) object encapsulates a provider-specific address.*
- class **ExceptionListener**  
*If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 1893) that is registered with the **Connection** (p. 1296).*

- class **IllegalStateException**

*This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.*

- class **InvalidClientIdException**

*This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.*

- class **InvalidDestinationException**

*This exception must be thrown when a destination either is not understood by a provider or is no longer valid.*

- class **InvalidSelectorException**

*This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.*

- class **MapMessage**

*A **MapMessage** (p. 2551) object is used to send a set of name-value pairs.*

- class **Message**

*Root of all messages.*

- class **MessageConsumer**

*A client uses a **MessageConsumer** (p. 2674) to received messages from a destination.*

- class **MessageEnumeration**

*Defines an object that enumerates a collection of Messages.*

- class **MessageEOFException**

*This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 3760) or **BytesMessage** (p. 1079) is being read.*

- class **MessageFormatException**

*This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.*

- class **MessageListener**

*A **MessageListener** (p. 2779) object is used to receive asynchronously delivered messages.*

- class **MessageNotReadableException**

*This exception must be thrown when a CMS client attempts to read a write-only message.*

- class **MessageNotWriteableException**

*This exception must be thrown when a CMS client attempts to write to a read-only message.*

- class **MessageProducer**

*A client uses a **MessageProducer** (p. 2809) object to send messages to a **Destination** (p. 1776).*

- class **ObjectMessage**

*Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.*

- class **Queue**

*An interface encapsulating a provider-specific queue name.*

- class **QueueBrowser**

*This class implements in interface for browsing the messages in a **Queue** (p. 3238) without removing them.*

- class **Session**

*A **Session** (p. 3460) object is a single-threaded context for producing and consuming messages.*

- class **Startable**

*Interface for a class that implements the start method.*

- class **Stoppable**

*Interface for a class that implements the stop method.*

- class **StreamMessage**

*Interface for a **StreamMessage** (p. 3760).*

- class **TemporaryQueue**

*Defines a Temporary **Queue** (p. 3238) based **Destination** (p. 1776).*

- class **TemporaryTopic**

*Defines a Temporary **Topic** (p. 3930) based **Destination** (p. 1776).*

- class **TextMessage**

*Interface for a text message.*

- class **Topic**

*An interface encapsulating a provider-specific topic name.*

- class **UnsupportedOperationException**

*This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.*

### 5.30.1 Detailed Description

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## 5.31 decaf Namespace Reference

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

### Namespaces

- namespace **internal**
- namespace **io**
- namespace **lang**
- namespace **net**
- namespace **nio**
- namespace **security**
- namespace **util**

### 5.31.1 Detailed Description

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## 5.32 decaf::internal Namespace Reference

### Namespaces

- namespace **io**
- namespace **net**
- namespace **nio**
- namespace **security**
- namespace **util**

### Data Structures

- class **AprPool**  
*Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.*
- class **DecafRuntime**  
*Handles APR initialization and termination.*

## 5.33 decaf::internal::io Namespace Reference

### Data Structures

- class **StandardErrorOutputStream**  
*Wrapper Around the Standard error Output facility on the current platform.*
- class **StandardInputStream**
- class **StandardOutputStream**

## 5.34 decaf::internal::net Namespace Reference

### Namespaces

- namespace **ssl**
- namespace **tcp**

### Data Structures

- class **DefaultServerSocketFactory**  
*Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.*

- class **DefaultSocketFactory**  
*SocketFactory implementation that is used to create Sockets.*
- class **Network**  
*Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.*
- class **SocketFileDescriptor**  
*File Descriptor type used internally by Decaf Socket objects.*
- class **URIEncoderDecoder**
- class **URIHelper**  
*Helper class used by the URI classes in encoding and decoding of URI's.*
- class **URIType**  
*Basic type object that holds data that composes a given URI.*

## 5.35 decaf::internal::net::ssl Namespace Reference

### Namespaces

- namespace **openssl**

### Data Structures

- class **DefaultSSLContext**  
*Default SSLContext manager for the Decaf library.*
- class **DefaultSSLServerSocketFactory**  
*Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.*
- class **DefaultSSLSocketFactory**  
*Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.*

## 5.36 decaf::internal::net::ssl::openssl Namespace Reference

### Data Structures

- class **OpenSSLContextSpi**



*Provides an SSLContext that wraps the OpenSSL API.*

- class **OpenSSLParameters**  
*Container class for parameters that are Common to OpenSSL socket classes.*
- class **OpenSSLServerSocket**  
*SSLServerSocket based on OpenSSL library code.*
- class **OpenSSLServerSocketFactory**  
*SSLServerSocketFactory that creates Server Sockets that use OpenSSL.*
- class **OpenSSLSocket**  
*Wraps a a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.*
- class **OpenSSLSocketException**  
*Subclass of the standard SocketException that knows how to produce an error message from the OpenSSL error stack.*
- class **OpenSSLSocketFactory**  
*Client Socket Factory that creates SSL based client sockets using the OpenSSL library.*
- class **OpenSSLSocketInputStream**  
*An output stream for reading data from an OpenSSL Socket instance.*
- class **OpenSSLSocketOutputStream**  
*OutputStream implementation used to write data to an **OpenSSLSocket** (p. 2940) instance.*

## 5.37 decaf::internal::net::tcp Namespace Reference

### Data Structures

- class **TcpSocket**  
*Platform-independent implementation of the socket interface.*
- class **TcpSocketInputStream**  
*Input stream for performing reads on a socket.*
- class **TcpSocketOutputStream**  
*Output stream for performing write operations on a socket.*

## 5.38 decaf::internal::nio Namespace Reference

### Data Structures

- class **BufferFactory**  
*Factory class used by static methods in the **decaf::nio** (p. 147) package to create the various default version of the NIO interfaces.*
- class **ByteBuffer**  
*This class defines six categories of operations upon byte buffers:*
  - class **CharArrayBuffer**
  - class **DoubleArrayBuffer**
  - class **FloatArrayBuffer**
  - class **IntArrayBuffer**
  - class **LongArrayBuffer**
  - class **ShortArrayBuffer**

## 5.39 decaf::internal::security Namespace Reference

### Data Structures

- class **SecureRandomImpl**  
*Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.*

## 5.40 decaf::internal::util Namespace Reference

### Namespaces

- namespace **concurrent**

### Data Structures

- class **ByteArrayAdapter**  
*This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.*
- class **GenericResource**  
*A Generic **Resource** (p. 3374) wraps some type and will delete it when the **Resource** (p. 3374) itself is deleted.*
- class **HexStringParser**

- class **Resource**

*Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.*

- class **ResourceLifecycleManager**
- class **TimerTaskHeap**

*A Binary Heap implemented specifically for the Timer class in Decaf Util.*

## 5.41 decaf::internal::util::concurrent Namespace Reference

### Data Structures

- class **ConditionImpl**
- class **MutexImpl**
- class **SynchronizableImpl**

*A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.*

- class **Transferer**

*Shared internal API for dual stacks and queues.*

- class **TransferQueue**

*This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.*

- class **TransferStack**

## 5.42 decaf::io Namespace Reference

### Data Structures

- class **BlockingByteArrayInputStream**

*This is a blocking version of a byte buffer stream.*

- class **BufferedInputStream**

*A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.*

- class **BufferedOutputStream**

*Wrapper around another output stream that buffers output before writing to the target output stream.*

- class **ByteArrayInputStream**

A ***ByteArrayInputStream*** (p. 1038) contains an internal buffer that contains bytes that may be read from the stream.

- class **ByteArrayOutputStream**
- class **Closeable**

*Interface for a class that implements the close method.*

- class **DataInput**

*The **DataInput** (p. 1603) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types.*

- class **DataInputStream**

*A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.*

- class **DataOutput**

*The **DataOutput** (p. 1622) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream.*

- class **DataOutputStream**

*A data output stream lets an application write primitive Java data types to an output stream in a portable way.*

- class **EOFException**

- class **FileDescriptor**

*This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.*

- class **FilterInputStream**

*A **FilterInputStream** (p. 1950) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.*

- class **FilterOutputStream**

*This class is the superclass of all classes that filter output streams.*

- class **Flushable**

*A **Flushable** (p. 1998) is a destination of data that can be flushed.*

- class **InputStream**

*A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.*

- class **InputStreamReader**

*An **InputStreamReader** (p. 2116) is a bridge from byte streams to character streams.*

- class **InterruptedIOException**

- class **IOException**

- class **OutputStream**  
*Base interface for any class that wants to represent an output stream of bytes.*
- class **OutputStreamWriter**  
*A class for turning a character stream into a byte stream.*
- class **PushbackInputStream**  
*A **PushbackInputStream** (p. 3231) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.*
- class **Reader**
- class **UnsupportedEncodingException**  
*Thrown when the the Character Encoding is not supported.*
- class **UTFDataFormatException**  
*Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.*
- class **Writer**

## 5.43 decaf::lang Namespace Reference

### Namespaces

- namespace **exceptions**

### Data Structures

- class **Appendable**  
*An object to which char sequences and values can be appended.*
- class **ArrayPointer**  
*Decaf's implementation of a Smart **Pointer** (p. 3032) that is a template on a Type and is **Thread** (p. 3876) Safe if the default Reference Counter is used.*
- class **ArrayPointerComparator**  
*This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this **ArrayPointer** (p. 736).*
- class **Boolean**
- class **Byte**
- class **Character**
- class **CharSequence**  
*A **CharSequence** (p. 1167) is a readable sequence of char values.*

- class **Comparable**

*This interface imposes a total ordering on the objects of each class that implements it.*

- class **Double**
- class **Exception**
- class **Float**
- class **Integer**
- class **Iterable**

*Implementing this interface allows an object to be cast to an **Iterable** (p. 2220) type for generic collections API calls.*

- class **Long**
- class **Math**

*The class **Math** (p. 2575) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.*

- class **Number**

*The abstract class **Number** (p. 2918) is the superclass of classes **Byte** (p. 969), **Double** (p. 1841), **Float** (p. 1961), **Integer** (p. 2143), **Long** (p. 2495), and **Short** (p. 3538).*

- struct **STATIC\_CAST\_TOKEN**
- struct **DYNAMIC\_CAST\_TOKEN**
- class **Pointer**

*Decaf's implementation of a Smart **Pointer** (p. 3032) that is a template on a Type and is **Thread** (p. 3876) Safe if the default Reference Counter is used.*

- class **PointerComparator**

*This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 3032) instance.*

- class **Readable**

*A **Readable** (p. 3251) is a source of characters.*

- class **Runnable**

*Interface for a runnable object - defines a task that can be run by a thread.*

- class **Runtime**
- class **Short**
- class **String**

*The **String** (p. 3775) class represents an immutable sequence of chars.*

- class **System**

The **System** (p.3838) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.

- class **Thread**

A **Thread** (p. 3876) is a concurrent unit of execution.

- class **ThreadGroup**

- class **Throwable**

This class represents an error that has occurred.

## Functions

- template<typename T, typename R, typename U >  
bool **operator==** (const **ArrayPointer**< T, R > &left, const U \*right)
- template<typename T, typename R, typename U >  
bool **operator==** (const U \*left, const **ArrayPointer**< T, R > &right)
- template<typename T, typename R, typename U >  
bool **operator!=** (const **ArrayPointer**< T, R > &left, const U \*right)
- template<typename T, typename R, typename U >  
bool **operator!=** (const U \*left, const **ArrayPointer**< T, R > &right)
- template<typename T, typename R, typename U >  
bool **operator==** (const **Pointer**< T, R > &left, const U \*right)
- template<typename T, typename R, typename U >  
bool **operator==** (const U \*left, const **Pointer**< T, R > &right)
- template<typename T, typename R, typename U >  
bool **operator!=** (const **Pointer**< T, R > &left, const U \*right)
- template<typename T, typename R, typename U >  
bool **operator!=** (const U \*left, const **Pointer**< T, R > &right)

### 5.43.1 Function Documentation

5.43.1.1 template<typename T, typename R, typename U > bool decaf::lang::operator!= (const **ArrayPointer**< T, R > &left, const U \*right ) [inline]

References decaf::lang::ArrayPointer< T, REFCOUNTER >::get().

5.43.1.2 template<typename T, typename R, typename U > bool decaf::lang::operator!= (const U \*left, const **ArrayPointer**< T, R > &right ) [inline]

References decaf::lang::ArrayPointer< T, REFCOUNTER >::get().

5.43.1.3 template<typename T, typename R, typename U > bool decaf::lang::operator!= (const U \*left, const **Pointer**< T, R > &right ) [inline]

References decaf::lang::Pointer< T, REFCOUNTER >::get().

**5.43.1.4** `template<typename T , typename R , typename U > bool decaf::lang::operator!= ( const Pointer< T, R > & left, const U * right ) [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

**5.43.1.5** `template<typename T , typename R , typename U > bool decaf::lang::operator== ( const Pointer< T, R > & left, const U * right ) [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

**5.43.1.6** `template<typename T , typename R , typename U > bool decaf::lang::operator== ( const U * left, const ArrayPointer< T, R > & right ) [inline]`

References `decaf::lang::ArrayPointer< T, REFCOUNTER >::get()`.

**5.43.1.7** `template<typename T , typename R , typename U > bool decaf::lang::operator== ( const U * left, const Pointer< T, R > & right ) [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

**5.43.1.8** `template<typename T , typename R , typename U > bool decaf::lang::operator== ( const ArrayPointer< T, R > & left, const U * right ) [inline]`

References `decaf::lang::ArrayPointer< T, REFCOUNTER >::get()`.

## 5.44 decaf::lang::exceptions Namespace Reference

### Data Structures

- class **ClassCastException**
- class **IllegalArgumentException**
- class **IllegalMonitorStateException**
- class **IllegalStateException**
- class **IllegalThreadStateException**
- class **IndexOutOfBoundsException**
- class **InterruptedException**
- class **InvalidStateException**
- class **NoSuchElementException**
- class **NullPointerException**
- class **NumberFormatException**
- class **RuntimeException**
- class **UnsupportedOperationException**



## 5.45 decaf::net Namespace Reference

### Namespaces

- namespace **ssl**

### Data Structures

- class **BindException**
- class **ConnectException**
- class **HttpRetryException**
- class **Inet4Address**
- class **Inet6Address**
- class **InetAddress**

*Represents an IP address.*

- class **InetSocketAddress**
- class **MalformedURLException**
- class **NoRouteToHostException**
- class **PortUnreachableException**
- class **ProtocolException**
- class **ServerSocket**

*This class implements server sockets.*

- class **ServerSocketFactory**

*Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.*

- class **Socket**
- class **SocketAddress**

*Base class for protocol specific **Socket** (p. 3607) addresses.*

- class **SocketError**

*Static utility class to simplify handling of error codes for socket operations.*

- class **SocketException**

*Exception for errors when manipulating sockets.*

- class **SocketFactory**

*The **SocketFactory** (p. 3629) is used to create **Socket** (p. 3607) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.*

- class **SocketImpl**

*Acts as a base class for all physical **Socket** (p. 3607) implementations.*

- class **SocketImplFactory**

*Factory class interface for a Factory that creates SocketImpl objects.*

- class **SocketOptions**
- class **SocketTimeoutException**
- class **UnknownHostException**
- class **UnknownServiceException**
- class **URI**

*This class represents an instance of a **URI** (p.4031) as defined by RFC 2396.*

- class **URISyntaxException**
- class **URL**

*Class **URL** (p.4072) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.*

- class **URLDecoder**
- class **URLEncoder**

## 5.46 decaf::net::ssl Namespace Reference

### Data Structures

- class **SSLContext**

*Represents an implementation of the Secure **Socket** (p.3607) Layer for streaming based sockets.*

- class **SSLContextSpi**

*Defines the interface that should be provided by an **SSLContext** (p.3653) provider.*

- class **SSLParameters**
- class **SSLServerSocket**

*Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol.*

- class **SSLServerSocketFactory**

*Factory class interface that provides methods to create SSL Server Sockets.*

- class **SSLSocket**
- class **SSLSocketFactory**

*Factory class interface for a **SocketFactory** (p.3629) that can create **SSLSocket** (p.3670) objects.*

## 5.47 decaf::nio Namespace Reference

### Data Structures

- class **Buffer**  
*A container for data of a specific primitive type.*
- class **BufferOverflowException**
- class **BufferUnderflowException**
- class **ByteBuffer**  
*This class defines six categories of operations upon byte buffers:*
- class **CharBuffer**  
*This class defines four categories of operations upon character buffers:*
- class **DoubleBuffer**  
*This class defines four categories of operations upon double buffers:*
- class **FloatBuffer**  
*This class defines four categories of operations upon float buffers:*
- class **IntBuffer**  
*This class defines four categories of operations upon int buffers:*
- class **InvalidMarkException**
- class **LongBuffer**  
*This class defines four categories of operations upon long long buffers:*
- class **ReadOnlyBufferException**
- class **ShortBuffer**  
*This class defines four categories of operations upon short buffers:*

## 5.48 decaf::security Namespace Reference

### Namespaces

- namespace **auth**
- namespace **cert**

### Data Structures

- class **GeneralSecurityException**
- class **InvalidKeyException**
- class **Key**

*The **Key** (p. 2364) interface is the top-level interface for all keys.*

- class **KeyException**
- class **KeyManagementException**
- class **NoSuchAlgorithmException**
- class **NoSuchProviderException**
- class **Principal**

*Base interface for a principal, which can represent an individual or organization.*

- class **PublicKey**

*A public key.*

- class **SecureRandom**
- class **SecureRandomSpi**

*Interface class used by Security Service Providers to implement a source of secure random bytes.*

- class **SignatureException**

## 5.49 decaf::security::auth Namespace Reference

### Namespaces

- namespace **x500**

## 5.50 decaf::security::auth::x500 Namespace Reference

### Data Structures

- class **X500Principal**

## 5.51 decaf::security::cert Namespace Reference

### Data Structures

- class **Certificate**  
*Base interface for all identity certificates.*
- class **CertificateEncodingException**
- class **CertificateException**
- class **CertificateExpiredException**
- class **CertificateNotYetValidException**
- class **CertificateParsingException**

- class **X509Certificate**  
*Base interface for all identity certificates.*

## 5.52 decaf::util Namespace Reference

### Namespaces

- namespace **comparators**
- namespace **concurrent**
- namespace **logging**
- namespace **zip**

### Data Structures

- class **AbstractCollection**  
*This class provides a skeletal implementation of the **Collection** (p. 1216) interface, to minimize the effort required to implement this interface.*
- class **AbstractList**  
*This class provides a skeletal implementation of the **List** (p. 2409) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).*
- class **AbstractMap**  
*This class provides a skeletal implementation of the **Map** (p. 2538) interface, to minimize the effort required to implement this interface.*
- class **AbstractQueue**  
*This class provides skeletal implementations of some **Queue** (p. 3239) operations.*
- class **AbstractSequentialList**  
*This class provides a skeletal implementation of the **List** (p. 2409) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).*
- class **AbstractSet**  
*This class provides a skeletal implementation of the **Set** (p. 3538) interface to minimize the effort required to implement this interface.*
- class **Collection**  
*The root interface in the collection hierarchy.*
- class **Comparator**  
*A comparison function, which imposes a total ordering on some collection of objects.*

- class **Date**  
*Wrapper class around a time value in milliseconds.*
- class **Iterator**  
*Defines an object that can be used to iterate over the elements of a collection.*
- class **List**  
*An ordered collection (also known as a sequence).*
- class **ListIterator**  
*An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.*
- class **Map**  
***Map** (p. 2538) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*
- class **PriorityQueue**  
*An unbounded priority queue based on a binary heap algorithm.*
- class **Properties**  
*Java-like properties class for mapping string names to string values.*
- class **Queue**  
*A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.*
- class **Random**  
***Random** (p. 3245) Value Generator which is used to generate a stream of pseudorandom numbers.*
- class **Set**  
*A collection that contains no duplicate elements.*
- class **StlList**  
***List** (p. 2409) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.*
- class **StlMap**  
***Map** (p. 2538) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*
- class **StlQueue**  
*The **Queue** (p. 3239) class accepts messages with an `psuh(m)` command where `m` is the message to be queued.*
- class **StlSet**

*Set* (p. 3538) template that wraps around a `std::set` to provide a more user-friendly interface and to provide common functions that do not exist in `std::set`.

- class **StringTokenizer**
- class **Timer**

*A facility for threads to schedule tasks for future execution in a background thread.*

- class **TimerTask**

*A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 3901).*

- class **UUID**

*A class that represents an immutable universally unique identifier (**UUID** (p. 4080)).*

## 5.53 `decaf::util::comparators` Namespace Reference

### Data Structures

- class **Less**

*Simple **Less** (p. 2399) **Comparator** (p. 1251) that compares to elements to determine if the first is less than the second.*

## 5.54 `decaf::util::concurrent` Namespace Reference

### Namespaces

- namespace **atomic**
- namespace **locks**

### Data Structures

- class **ConditionHandle**
- class **MutexHandle**
- class **BlockingQueue**

*A `decaf::util::Queue` (p. 3239) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.*

- class **BrokenBarrierException**
- class **Callable**

*A task that returns a result and may throw an exception.*

- class **CancellationException**
- class **ConcurrentMap**  
*Interface for a **Map** (p. 2538) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 2538) interface.*
- class **ConcurrentStlMap**  
***Map** (p. 2538) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.*
- class **CountDownLatch**
- class **Delayed**  
*A mix-in style interface for marking objects that should be acted upon after a given delay.*
- class **ExecutionException**
- class **Executor**  
*An object that executes submitted **decaf.lang Runnable** (p. 3418) tasks.*
- class **ExecutorService**  
*An **Executor** (p. 1926) that provides methods to manage termination and methods that can produce a **Future** (p. 2029) for tracking progress of one or more asynchronous tasks.*
- class **Future**  
*A **Future** (p. 2029) represents the result of an asynchronous computation.*
- class **Lock**  
*A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.*
- class **Mutex**  
***Mutex** (p. 2866) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.*
- class **PooledThread**
- class **PooledThreadListener**  
*Abstract Listener Interface for users of **ThreadPool** (p. 3888).*
- class **RejectedExecutionException**
- class **RejectedExecutionHandler**  
*A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. ??).*
- class **Semaphore**  
*A counting semaphore.*
- class **Synchronizable**  
*The interface for all synchronizable objects (that is, objects that can be locked and unlocked).*



- class **SynchronousQueue**

A **blocking queue** (p. 848) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.

- class **TaskListener**
- class **ThreadFactory**

public interface **ThreadFactory** (p. 3887)

- class **ThreadPool**

Defines a **Thread Pool** object that implements the functionality of pooling threads to perform user tasks.

- class **TimeoutException**
- class **TimeUnit**

A **TimeUnit** (p. 3919) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.

## 5.55 decaf::util::concurrent::atomic Namespace Reference

### Data Structures

- class **AtomicBoolean**

A boolean value that may be updated atomically.

- class **AtomicInteger**

An int value that may be updated atomically.

- class **AtomicRefCounter**
- class **AtomicReference**

An **Pointer** reference that may be updated atomically.

## 5.56 decaf::util::concurrent::locks Namespace Reference

### Data Structures

- class **Condition**

**Condition** (p. 1282) factors out the **Mutex** (p. 2866) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 2452) implementations.

- class **Lock**

***Lock** (p. 2452) implementations provide more extensive locking operations than can be obtained using synchronized statements.*

- class **LockSupport**

*Basic thread blocking primitives for creating locks and other synchronization classes.*

- class **ReadWriteLock**

*A **ReadWriteLock** (p. 3263) maintains a pair of associated locks, one for read-only operations and one for writing.*

- class **ReentrantLock**

*A reentrant mutual exclusion **Lock** (p. 2452) with extended capabilities.*

## 5.57 decaf::util::logging Namespace Reference

### Data Structures

- class **ConsoleHandler**

*This **Handler** (p. 2042) publishes log records to `System.err`.*

- class **ErrorManager**

***ErrorManager** (p. 1884) objects can be attached to **Handlers** to process any error that occur on a **Handler** (p. 2042) during Logging.*

- class **Filter**

*A **Filter** (p. 1949) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.*

- class **Formatter**

*A **Formatter** (p. 2027) provides support for formatting **LogRecords**.*

- class **Handler**

*A **Handler** (p. 2042) object takes log messages from a **Logger** (p. 2461) and exports them.*

- class **Level**

*The **Level** (p. 2403) class defines a set of standard logging levels that can be used to control logging output.*

- class **Logger**

*A **Logger** (p. 2461) object is used to log messages for a specific system or application component.*

- class **LoggerHierarchy**

- class **LogManager**

There is a single global **LogManager** (p. 2480) object that is used to maintain a set of shared state about Loggers and log services.

- class **LogRecord**

**LogRecord** (p. 2487) objects are used to pass logging requests between the logging framework and individual log Handlers.

- class **LogWriter**

- class **MarkBlockLogger**

Defines a class that can be used to mark the entry and exit from scoped blocks.

- class **PropertiesChangeListener**

Defines the interface that classes can use to listen for change events on **Properties** (p. 3216).

- class **SimpleFormatter**

Print a brief summary of the **LogRecord** (p. 2487) in a human readable format.

- class **SimpleLogger**

- class **StreamHandler**

Stream based logging **Handler** (p. 2042).

- class **XMLFormatter**

Format a **LogRecord** (p. 2487) into a standard XML format.

## Enumerations

- enum **Levels** {

**Off**, **Null**, **Markblock**, **Debug**,

**Info**, **Warn**, **Error**, **Fatal**,

**Throwing** }

Defines an enumeration for logging levels.

### 5.57.1 Enumeration Type Documentation

#### 5.57.1.1 enum decaf::util::logging::Levels

Defines an enumeration for logging levels.

#### Enumerator:

*Off*

*Null*

*Markblock*

*Debug*  
*Info*  
*Warn*  
*Error*  
*Fatal*  
*Throwing*

## 5.58 decaf::util::zip Namespace Reference

### Data Structures

- class **Adler32**  
*Clas that can be used to compute an Adler-32 **Checksum** (p. 1174) for a data stream.*
- class **CheckedInputStream**  
*An implementation of a **FilterInputStream** that will maintain a **Checksum** (p. 1174) of the bytes read, the **Checksum** (p. 1174) can then be used to verify the integrity of the input stream.*
- class **CheckedOutputStream**  
*An implementation of a **FilterOutputStream** that will maintain a **Checksum** (p. 1174) of the bytes written, the **Checksum** (p. 1174) can then be used to verify the integrity of the output stream.*
- class **Checksum**  
*An interface used to represent **Checksum** (p. 1174) values in the Zip package.*
- class **CRC32**  
*Class that can be used to compute a CRC-32 checksum for a data stream.*
- class **DataFormatException**
- class **Deflater**  
*This class compresses data using the DEFLATE algorithm (see specification).*
- class **DeflaterOutputStream**  
*Provides a **FilterOutputStream** instance that compresses the data before writing it to the wrapped **OutputStream**.*
- class **Inflater**  
*This class uncompresses data that was compressed using the DEFLATE algorithm (see specification).*
- class **InflaterInputStream**  
*A **FilterInputStream** that decompresses data read from the wrapped **InputStream** instance.*
- class **ZipException**

## 5.59 std Namespace Reference

### Data Structures

- struct **less**< **decaf::lang::ArrayPointer**< **T** > >  
*An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.*
- struct **less**< **decaf::lang::Pointer**< **T** > >  
*An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.*



## Chapter 6

# Data Structure Documentation

### 6.1 decaf::util::AbstractCollection< E > Class Template Reference

This class provides a skeletal implementation of the **Collection** (p. 1216) interface, to minimize the effort required to implement this interface.

```
#include <src/main/decaf/util/AbstractCollection.h>
```

Inheritance diagram for decaf::util::AbstractCollection< E >:

#### Public Member Functions

- **AbstractCollection** ()
- virtual ~**AbstractCollection** ()
- **AbstractCollection**< E > & **operator=** (const **AbstractCollection**< E > &collection)  
*Assignment Operator, copy element from the source collection to this collection after clearing any element stored in this collection.*
- virtual bool **add** (const E &value DECAF\_UNUSED) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )  
*Ensures that this collection contains the specified element (optional operation).*
- virtual bool **addAll** (const **Collection**< E > &collection) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )  
*Adds all of the elements in the specified collection to this collection (optional operation).*
- virtual void **clear** () throw ( lang::exceptions::UnsupportedOperationException )

*Removes all of the elements from this collection (optional operation).*

- virtual void **copy** (const **Collection**< E > &collection)

*Renders this **Collection** (p. 1216) as a Copy of the given **Collection** (p. 1216).*

- virtual bool **contains** (const E &value) const throw ( lang::Exception )

*Returns true if this collection contains the specified element.*

- virtual bool **containsAll** (const **Collection**< E > &collection) const throw ( lang::Exception )

*Returns true if this collection contains all of the elements in the specified collection.*

- virtual bool **equals** (const **Collection**< E > &collection) const

*Answers true if this **Collection** (p. 1216) and the one given are the same size and if each element contained in the **Collection** (p. 1216) given is equal to an element contained in this collection.*

- virtual bool **isEmpty** () const

*Returns true if this collection contains no elements.*

- virtual bool **remove** (const E &value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )

*Removes a single instance of the specified element from this collection, if it is present (optional operation).*

- virtual bool **removeAll** (const **Collection**< E > &collection) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )

*Removes all of this collection's elements that are also contained in the specified collection (optional operation).*

- virtual bool **retainAll** (const **Collection**< E > &collection) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )

*Retains only the elements in this collection that are contained in the specified collection (optional operation).*

- virtual std::vector< E > **toArray** () const

*Answers an STL vector containing copies of all elements contained in this **Collection** (p. 1216).*

- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )

*Locks the object.*

- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )

*Attempts to Lock the object, if the lock is already held by another thread than this method returns false.*

- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )

*Unlocks the object.*



- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals a waiter on this object that it can now wake up and continue.*

- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals the waiters on this object that it can now wake up and continue.*

## Protected Attributes

- **util::concurrent::Mutex mutex**

### 6.1.1 Detailed Description

**template<typename E> class decaf::util::AbstractCollection< E >**

This class provides a skeletal implementation of the **Collection** (p. 1216) interface, to minimize the effort required to implement this interface. To implement an unmodifiable collection, the programmer needs only to extend this class and provide implementations for the iterator and size methods. (The iterator returned by the iterator method must implement hasNext and next.)

To implement a modifiable collection, the programmer must additionally override this class's add method (which otherwise throws an UnsupportedOperationException), and the iterator returned by the iterator method must additionally implement its remove method.

The programmer should generally provide a void (no argument) and **Collection** (p. 1216) constructor, as per the recommendation in the **Collection** (p. 1216) interface specification.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the collection being implemented admits a more efficient implementation.

### Since

1.0

## 6.1.2 Constructor & Destructor Documentation

6.1.2.1 `template<typename E> decaf::util::AbstractCollection< E  
>::AbstractCollection ( ) [inline]`

6.1.2.2 `template<typename E> virtual decaf::util::AbstractCollection< E  
>::~~AbstractCollection ( ) [inline, virtual]`

## 6.1.3 Member Function Documentation

6.1.3.1 `template<typename E> virtual bool decaf::util::AbstractCollection<  
E >::add ( const E &value DECAF_UNUSED ) throw (  
lang::exceptions::UnsupportedOperationException,  
lang::exceptions::IllegalArgumentException,  
lang::exceptions::IllegalStateException ) [inline, virtual]`

Ensures that this collection contains the specified element (optional operation).

Returns true if this collection changed as a result of the call. (Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. **Collection** (p. 1216) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

This implementation always throws an UnsupportedOperationException.

### Parameters

<i>value</i>	- The element that must be ensured to be in this collection.
--------------	--

### Returns

true if the collection was changed as a result of this call.

### Exceptions

<i>UnsupportedOperationException</i>	if the add operation is not supported by this collection
--------------------------------------	--

<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions

Referenced by decaf::util::AbstractCollection< cms::Connection \* >::addAll(), decaf::util::AbstractCollection< cms::Connection \* >::copy(), and decaf::util::AbstractCollection< cms::Connection \* >::operator=().

**6.1.3.2** `template<typename E> virtual bool decaf::util::AbstractCollection< E >::addAll ( const Collection< E > & collection ) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException ) [inline, virtual]`

Adds all of the elements in the specified collection to this collection (optional operation).

The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an UnsupportedOperationException unless add is overridden (assuming the specified collection is non-empty).

#### Parameters

<i>collection</i>	- The <b>Collection</b> (p. 1216) whose elements are to be added to this <b>Collection</b> (p. 1216).
-------------------	---

#### Returns

true if the collection was changed as a result of this call.

#### Exceptions

<i>UnsupportedOperationException</i>	if the addAll operation is not supported by this collection
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions

Implements **decaf::util::Collection< E >** (p. 1219).

Reimplemented in **decaf::util::AbstractQueue< E >** (p. 177).

```
6.1.3.3 template<typename E> virtual void decaf::util::AbstractCollection< E >::clear (
    ) throw ( lang::exceptions::UnsupportedOperationException ) [inline,
    virtual]
```

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 2223) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

### Exceptions

<i>UnsupportedOperationException</i>	if the clear operation is not supported by this collection
--------------------------------------	--

Implements **decaf::util::Collection< E >** (p. 1220).

Reimplemented in **decaf::util::AbstractQueue< E >** (p. 178), **decaf::util::concurrent::SynchronousQueue< E >** (p. 3830), **decaf::util::PriorityQueue< E >** (p. 3120), **decaf::util::StlList< E >** (p. 3702), **decaf::util::StlSet< E >** (p. 3735), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3702), **decaf::util::StlList< CompositeTask \* >** (p. 3702), **decaf::util::StlList< URI >** (p. 3702), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3702), **decaf::util::StlList< PrimitiveValueNode >** (p. 3702), **decaf::util::StlList< Pointer< Command > >** (p. 3702), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3702), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3702), **decaf::util::StlList< cms::Destination \* >** (p. 3702), **decaf::util::StlList< cms::Session \* >** (p. 3702), **decaf::util::StlList< cms::Connection \* >** (p. 3702), **decaf::util::StlSet< transport::TransportListener \* >** (p. 3735), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 3735), **decaf::util::StlSet< Resource \* >** (p. 3735), and **decaf::util::StlSet< ActiveMQSession \* >** (p. 3735).

Referenced by **decaf::util::AbstractCollection< cms::Connection \* >::copy()**, and **decaf::util::AbstractCollection< cms::Connection \* >::operator=()**.

```
6.1.3.4 template<typename E> virtual bool decaf::util::AbstractCollection< E
    >::contains ( const E & value ) const throw ( lang::Exception ) [inline,
    virtual]
```

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

### Parameters

<i>value</i>	- the value whose presence is to be queried for in this <b>Collection</b> (p. 1216).
--------------	--

**Returns**

true if the value is contained in this collection

**Exceptions**

<i>Exception</i>	if an error occurs,
------------------	---------------------

Implements **decaf::util::Collection< E >** (p. 1221).

Reimplemented in **decaf::util::StlList< E >** (p. 3702), **decaf::util::StlSet< E >** (p. 3736), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3702), **decaf::util::StlList< CompositeTask \* >** (p. 3702), **decaf::util::StlList< URI >** (p. 3702), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3702), **decaf::util::StlList< PrimitiveValueNode >** (p. 3702), **decaf::util::StlList< Pointer< Command > >** (p. 3702), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3702), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3702), **decaf::util::StlList< cms::Destination \* >** (p. 3702), **decaf::util::StlList< cms::Session \* >** (p. 3702), **decaf::util::StlList< cms::Connection \* >** (p. 3702), **decaf::util::StlSet< transport::TransportListener \* >** (p. 3736), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 3736), **decaf::util::StlSet< Resource \* >** (p. 3736), and **decaf::util::StlSet< ActiveMQSession \* >** (p. 3736).

Referenced by **decaf::util::AbstractCollection< cms::Connection \* >::containsAll()**.

**6.1.3.5** `template<typename E> virtual bool decaf::util::AbstractCollection< E >::containsAll ( const Collection< E > & collection ) const throw ( lang::Exception ) [inline, virtual]`

Returns true if this collection contains all of the elements in the specified collection.

This implementation iterates over the specified collection, checking each element returned by the iterator in turn to see if it's contained in this collection. If all elements are so contained true is returned, otherwise false.

**Parameters**

<i>collection</i>	collection to be checked for containment in this collection
-------------------	---

**Returns**

true if this collection contains all of the elements in the specified collection.

**Exceptions**

<i>Exception</i>	if an error occurs,
------------------	---------------------

Implements **decaf::util::Collection< E >** (p. 1222).

Reimplemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3830).

Referenced by **decaf::util::AbstractCollection< cms::Connection \* >::equals()**.

**6.1.3.6** `template<typename E> virtual void decaf::util::AbstractCollection< E >::copy ( const Collection< E > & collection ) [inline, virtual]`

Renders this **Collection** (p. 1216) as a Copy of the given **Collection** (p. 1216).

This implementation iterates over the contents of the given collection adding each to this collection after first calling this Collection's clear method.

#### Parameters

<i>collection</i>	- the collection to mirror.
-------------------	-----------------------------

**6.1.3.7** `template<typename E> virtual bool decaf::util::AbstractCollection< E >::equals ( const Collection< E > & collection ) const [inline, virtual]`

Answers true if this **Collection** (p. 1216) and the one given are the same size and if each element contained in the **Collection** (p. 1216) given is equal to an element contained in this collection.

#### Parameters

<i>collection</i>	- The <b>Collection</b> (p. 1216) to be compared to this one.
-------------------	---

#### Returns

true if this **Collection** (p. 1216) is equal to the one given.

Implements **decaf::util::Collection< E >** (p. 1222).

Reimplemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3832).

**6.1.3.8** `template<typename E> virtual bool decaf::util::AbstractCollection< E >::isEmpty ( ) const [inline, virtual]`

Returns true if this collection contains no elements.

This implementation returns **size()** (p. 1226) == 0.

#### Returns

true if the size method return 0.

Implements **decaf::util::Collection< E >** (p. 1223).

Reimplemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3833), **decaf::util::StlList< E >** (p. 3704), **decaf::util::StlSet< E >** (p. 3736), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3704), **decaf::util::StlList< CompositeTask \* >** (p. 3704), **decaf::util::StlList< URI >** (p. 3704), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3704), **decaf::util::StlList< PrimitiveValueNode >** (p. 3704), **decaf::util::StlList< Pointer< Command > >** (p. 3704), **decaf::util::StlList< Pointer<**

**BackupTransport** > > (p. 3704), **decaf::util::StlList**< **cms::MessageProducer** \* > (p. 3704), **decaf::util::StlList**< **cms::Destination** \* > (p. 3704), **decaf::util::StlList**< **cms::Session** \* > (p. 3704), **decaf::util::StlList**< **cms::Connection** \* > (p. 3704), **decaf::util::StlSet**< **transport::TransportListener** \* > (p. 3736), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 3736), **decaf::util::StlSet**< **Resource** \* > (p. 3736), and **decaf::util::StlSet**< **ActiveMQSession** \* > (p. 3736).

Referenced by **decaf::util::AbstractQueue**< **E** >::clear().

**6.1.3.9** `template<typename E> virtual void decaf::util::AbstractCollection< E >::lock  
( ) throw ( decaf::lang::exceptions::RuntimeException ) [inline,  
virtual]`

Locks the object.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3812).

**6.1.3.10** `template<typename E> virtual void decaf::util::AbstractCollection<  
E >::notify ( ) throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalMonitorStateException ) [inline,  
virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

#### Exceptions

<i>IllegalMonitorState- Exception</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3813).

**6.1.3.11** `template<typename E> virtual void decaf::util::AbstractCollection< E  
>::notifyAll ( ) throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalMonitorStateException ) [inline,  
virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

#### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p.3814).

**6.1.3.12** `template<typename E> AbstractCollection<E>&  
decaf::util::AbstractCollection< E >::operator= ( const  
AbstractCollection< E > & collection ) [inline]`

Assignment Operator, copy element from the source collection to this collection after clearing any element stored in this collection.

#### Parameters

<i>collection</i>	- the collection to copy
-------------------	--------------------------

#### Returns

a reference to this collection

**6.1.3.13** `template<typename E> virtual bool decaf::util::AbstractCollection<  
E >::remove ( const E & value ) throw (  
lang::exceptions::UnsupportedOperationException,  
lang::exceptions::IllegalArgumentException ) [inline, virtual]`

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

#### Parameters

<i>value</i>	- element to be removed from this collection, if present
--------------	--

#### Returns

true if an element was removed as a result of this call



**Exceptions**

<i>UnsupportedOperationException</i>	if the remove operation is not supported by this collection.
<i>IllegalArgumentException</i>	If the value is not a valid entry for this <b>Collection</b> (p. 1216).

Implements **decaf::util::Collection< E >** (p. 1223).

Reimplemented in **decaf::util::PriorityQueue< E >** (p. 3123), **decaf::util::StlList< E >** (p. 3706), **decaf::util::StlSet< E >** (p. 3737), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3706), **decaf::util::StlList< CompositeTask \* >** (p. 3706), **decaf::util::StlList< URI >** (p. 3706), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3706), **decaf::util::StlList< PrimitiveValueNode >** (p. 3706), **decaf::util::StlList< Pointer< Command > >** (p. 3706), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3706), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3706), **decaf::util::StlList< cms::Destination \* >** (p. 3706), **decaf::util::StlList< cms::Session \* >** (p. 3706), **decaf::util::StlList< cms::Connection \* >** (p. 3706), **decaf::util::StlSet< transport::TransportListener \* >** (p. 3737), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 3737), **decaf::util::StlSet< Resource \* >** (p. 3737), and **decaf::util::StlSet< ActiveMQSession \* >** (p. 3737).

```
6.1.3.14  template<typename E> virtual bool decaf::util::AbstractCollection<
           E >::removeAll ( const Collection< E > & collection ) throw
           ( lang::exceptions::UnsupportedOperationException,
             lang::exceptions::IllegalArgumentException ) [inline, virtual]
```

Removes all of this collection's elements that are also contained in the specified collection (optional operation).

After this call returns, this collection will contain no elements in common with the specified collection.

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements in common with the specified collection.

**Parameters**

<i>collection</i>	- collection containing elements to be removed from this collection
-------------------	---

**Returns**

true if this collection changed as a result of the call

**Exceptions**

<i>UnsupportedOperationException</i>	if the remove operation is not supported by this collection
--------------------------------------	---

<i>IllegalArgumentException.</i>	
----------------------------------	--

Implements **decaf::util::Collection< E >** (p. 1224).

Reimplemented in **decaf::util::AbstractSet< E >** (p. 181), **decaf::util::AbstractSet< transport::TransportListener \* >** (p. 181), **decaf::util::AbstractSet< Pointer< Synchronization > >** (p. 181), **decaf::util::AbstractSet< Resource \* >** (p. 181), and **decaf::util::AbstractSet< ActiveMQSession \* >** (p. 181).

**6.1.3.15** `template<typename E> virtual bool decaf::util::AbstractCollection< E >::retainAll ( const Collection< E > & collection ) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException ) [inline, virtual]`

Retains only the elements in this collection that are contained in the specified collection (optional operation).

In other words, removes from this collection all of its elements that are not contained in the specified collection.

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's not so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements not present in the specified collection.

#### Parameters

<i>collection</i>	- collection containing elements to be retained in this collection
-------------------	--

#### Returns

true if this collection changed as a result of the call

#### Exceptions

<i>UnsupportedOperationException</i>	if the remove operation is not supported by this collection
<i>IllegalArgumentException.</i>	

Implements **decaf::util::Collection< E >** (p. 1225).

```
6.1.3.16  template<typename E> virtual std::vector<E>
          decaf::util::AbstractCollection< E >::toArray ( ) const  [inline,
          virtual]
```

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 1216).

All the elements in the array will not be referenced by the collection. The elements in the returned array will be sorted to the same order as those returned by the iterator of this collection itself if the collection guarantees the order.

#### Returns

an vector of copies of all the elements from this **Collection** (p. 1216)

Implements **decaf::util::Collection< E >** (p. 1226).

Reimplemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3838).

```
6.1.3.17  template<typename E> virtual bool decaf::util::AbstractCollection< E
          >::tryLock ( ) throw ( decaf::lang::exceptions::RuntimeException )
          [inline, virtual]
```

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

#### Returns

true if the lock was acquired, false if it is already held by another thread.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3815).

```
6.1.3.18  template<typename E> virtual void decaf::util::AbstractCollection< E
          >::unlock ( ) throw ( decaf::lang::exceptions::RuntimeException )
          [inline, virtual]
```

Unlocks the object.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3816).

```
6.1.3.19 template<typename E> virtual void decaf::util::AbstractCollection<
    E >::wait ( ) throw ( decaf::lang::exceptions::RuntimeException,
    decaf::lang::exceptions::IllegalMonitorStateException,
    decaf::lang::exceptions::InterruptedException ) [inline,
    virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3818).

```
6.1.3.20 template<typename E> virtual void decaf::util::AbstractCollection<
    E >::wait ( long long millisecs, int nanos ) throw
    ( decaf::lang::exceptions::RuntimeException,
    decaf::lang::exceptions::IllegalArgumentException,
    decaf::lang::exceptions::IllegalMonitorStateException,
    decaf::lang::exceptions::InterruptedException ) [inline,
    virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

#### Exceptions

<i>IllegalArgumentException</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3820).

```
6.1.3.21  template<typename E> virtual void decaf::util::AbstractCollection< E >::wait
( long long millisecs ) throw ( decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException ) [inline,
virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3819).

### 6.1.4 Field Documentation

```
6.1.4.1  template<typename E> util::concurrent::Mutex
decaf::util::AbstractCollection< E >::mutex [mutable,
protected]
```

Referenced by `decaf::util::AbstractCollection< cms::Connection * >::lock()`, `decaf::util::AbstractCollection< cms::Connection * >::notify()`, `decaf::util::AbstractCollection< cms::Connection * >::notifyAll()`, `decaf::util::AbstractCollection< cms::Connection * >::tryLock()`, `decaf::util::AbstractCollection< cms::Connection * >::unlock()`, and `decaf::util::AbstractCollection< cms::Connection * >::wait()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractCollection.h`

## 6.2 decaf::util::AbstractList< E > Class Template Reference

This class provides a skeletal implementation of the **List** (p. 2409) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).

```
#include <src/main/decaf/util/AbstractList.h>
```

Inheritance diagram for `decaf::util::AbstractList< E >`:

## Public Member Functions

- virtual `~AbstractList()`

### 6.2.1 Detailed Description

`template<typename E> class decaf::util::AbstractList< E >`

This class provides a skeletal implementation of the **List** (p. 2409) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array). For sequential access data (such as a linked list), **AbstractSequentialList** (p. 179) should be used in preference to this class.

To implement an unmodifiable list, the programmer needs only to extend this class and provide implementations for the `get(int)` and `size()` (p. 1226) methods.

To implement a modifiable list, the programmer must additionally override the `set(int, E)` method (which otherwise throws an `UnsupportedOperationException`). If the list is variable-size the programmer must additionally override the `add(int, E)` and `remove(int)` methods.

The programmer should generally provide a void (no argument) and collection constructor, as per the recommendation in the **Collection** (p. 1216) interface specification.

Unlike the other abstract collection implementations, the programmer does not have to provide an iterator implementation; the iterator and list iterator are implemented by this class, on top of the "random access" methods: `get(int)`, `set(int, E)`, `add(int, E)` and `remove(int)`.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the collection being implemented admits a more efficient implementation.

#### Since

1.0

### 6.2.2 Constructor & Destructor Documentation

**6.2.2.1** `template<typename E> virtual decaf::util::AbstractList< E >::~~AbstractList()` [`inline`, `virtual`]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractList.h`

## 6.3 decaf::util::AbstractMap< K, V, COMPARATOR > Class Template Reference

This class provides a skeletal implementation of the **Map** (p. 2538) interface, to minimize the effort required to implement this interface.

```
#include <src/main/decaf/util/AbstractMap.h>
```

Inheritance diagram for decaf::util::AbstractMap< K, V, COMPARATOR >:

### Public Member Functions

- virtual  $\sim$ **AbstractMap** ()

#### 6.3.1 Detailed Description

```
template<typename K, typename V, typename COMPARATOR> class decaf::util::AbstractMap<  
K, V, COMPARATOR >
```

This class provides a skeletal implementation of the **Map** (p. 2538) interface, to minimize the effort required to implement this interface. To implement an unmodifiable map, the programmer needs only to extend this class and provide an implementation for the `entrySet` method, which returns a set-view of the map's mappings. Typically, the returned set will, in turn, be implemented atop **AbstractSet** (p. 180). This set should not support the `add` or `remove` methods, and its iterator should not support the `remove` method.

To implement a modifiable map, the programmer must additionally override this class's `put` method (which otherwise throws an `UnsupportedOperationException`), and the iterator returned by `entrySet().iterator()` must additionally implement its `remove` method.

The programmer should generally provide a void (no argument) and map constructor, as per the recommendation in the **Map** (p. 2538) interface specification.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the map being implemented admits a more efficient implementation.

### Since

1.0

### 6.3.2 Constructor & Destructor Documentation

6.3.2.1 `template<typename K , typename V , typename COMPARATOR > virtual  
decaf::util::AbstractMap< K, V, COMPARATOR >::~~AbstractMap ( )  
[inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractMap.h`

## 6.4 decaf::util::AbstractQueue< E > Class Template Reference

This class provides skeletal implementations of some **Queue** (p. 3239) operations.

```
#include <src/main/decaf/util/AbstractQueue.h>
```

Inheritance diagram for `decaf::util::AbstractQueue< E >`:

### Public Member Functions

- **AbstractQueue** ()
- virtual **~AbstractQueue** ()
- virtual bool **add** (const E &value) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an IllegalStateException if no space is currently available.*
- virtual bool **addAll** (const **Collection**< E > &collection) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )  
*Adds all the elements of a collection to the queue.*
- virtual E **remove** () throw ( decaf::lang::exceptions::NoSuchElementException )  
*Retrieves and removes the head of this queue.*
- virtual E **element** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Retrieves, but does not remove, the head of this queue.*
- virtual void **clear** () throw ( lang::exceptions::UnsupportedOperationException )  
*Removes all elements of the queue.*



### 6.4.1 Detailed Description

template<typename E> class decaf::util::AbstractQueue< E >

This class provides skeletal implementations of some **Queue** (p. 3239) operations. Methods add, remove, and element are based on offer, poll, and peek, respectively.

A **Queue** (p. 3239) implementation that extends this class must minimally define a method **Queue** (p. 3239). **offer(E)** which does not permit insertion of null elements, along with methods **Queue** (p. 3239). **peek()** (p. 3242), **Queue.poll()** (p. 3242), **Collection.size()** (p. 1226), and a **Collection.iterator()** (p. 2221) supporting **Iterator.remove()** (p. 2223). Typically, additional methods will be overridden as well. If these requirements cannot be met, consider instead subclassing **AbstractCollection** (p. 159).

Since

1.0

### 6.4.2 Constructor & Destructor Documentation

6.4.2.1 template<typename E> decaf::util::AbstractQueue< E >::AbstractQueue ( ) [inline]

6.4.2.2 template<typename E> virtual decaf::util::AbstractQueue< E >::~~AbstractQueue ( ) [inline, virtual]

### 6.4.3 Member Function Documentation

6.4.3.1 template<typename E> virtual bool decaf::util::AbstractQueue< E >::add ( const E & *value* ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException ) [inline, virtual]

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an IllegalStateException if no space is currently available.

This implementation returns true if offer succeeds, else throws an IllegalStateException.

#### Parameters

<i>value</i>	- the element to offer to the <b>Queue</b> (p. 3239).
--------------	---

#### Returns

true if the add succeeds.

#### Exceptions

<i>IllegalArgumentException</i>	if the element cannot be added.
---------------------------------	---------------------------------

Implements **decaf::util::Collection< E >** (p. 1218).

Reimplemented in **decaf::util::PriorityQueue< E >** (p. 3120).

References **decaf::util::Queue< E >::offer()**.

**6.4.3.2** `template<typename E> virtual bool decaf::util::AbstractQueue< E >::addAll ( const Collection< E > & collection ) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException ) [inline, virtual]`

Adds all the elements of a collection to the queue.

If the collection is the queue itself, then an `IllegalArgumentException` will be thrown out. If during the process, some runtime exception is thrown out, then part of the elements in the collection that have successfully added will remain in the queue.

The result of the method is undefined if the collection is modified during the process of the method.

#### Parameters

<i>collection</i>	- the collection to be added to the queue.
-------------------	--

#### Returns

true if the operation succeeds.

#### Exceptions

<i>IllegalArgumentException</i>	If the collection to be added to the queue is the queue itself.
---------------------------------	---

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 163).

**6.4.3.3** `template<typename E> virtual void decaf::util::AbstractQueue< E >::clear ( ) throw ( lang::exceptions::UnsupportedOperationException ) [inline, virtual]`

Removes all elements of the queue.

This implementation repeatedly invokes `poll` until it returns the empty marker.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 163).

Reimplemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3830), and **decaf::util::PriorityQueue< E >** (p. 3120).

References `decaf::util::AbstractCollection< E >::isEmpty()`, and `decaf::util::Queue< E >::poll()`.

**6.4.3.4** `template<typename E> virtual E decaf::util::AbstractQueue< E >::element  
( ) const throw ( decaf::lang::exceptions::NoSuchElementException )  
[inline, virtual]`

Retrieves, but does not remove, the head of this queue.

This method differs from `peek` only in that it throws an exception if this queue is empty.

This implementation returns the result of `peek` unless the queue is empty.

#### Returns

the element in the head of the queue.

#### Exceptions

<i>NoSuchElementException</i>	if the queue is empty.
-------------------------------	------------------------

Implements `decaf::util::Queue< E >` (p. 3241).

References `decaf::util::Queue< E >::peek()`.

Referenced by `decaf::util::concurrent::SynchronousQueue< E >::drainTo()`.

**6.4.3.5** `template<typename E> virtual E decaf::util::AbstractQueue< E >::remove ( )  
throw ( decaf::lang::exceptions::NoSuchElementException ) [inline,  
virtual]`

Retrieves and removes the head of this queue.

This method differs from `poll` only in that it throws an exception if this queue is empty.

This implementation returns the result of `poll` unless the queue is empty.

#### Returns

a copy of the element in the head of the queue.

#### Exceptions

<i>NoSuchElementException</i>	if the queue is empty.
-------------------------------	------------------------

Implements `decaf::util::Queue< E >` (p. 3243).

Reimplemented in `decaf::util::PriorityQueue< E >` (p. 3123).

References `decaf::util::Queue< E >::poll()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractQueue.h`

## 6.5 `decaf::util::AbstractSequentialList< E >` Class Template Reference

This class provides a skeletal implementation of the **List** (p. 2409) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).

```
#include <src/main/decaf/util/AbstractSequentialList.h>
```

Inheritance diagram for `decaf::util::AbstractSequentialList< E >`:

### Public Member Functions

- `virtual ~AbstractSequentialList ()`

#### 6.5.1 Detailed Description

```
template<typename E> class decaf::util::AbstractSequentialList< E >
```

This class provides a skeletal implementation of the **List** (p. 2409) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list). For random access data (such as an array), **AbstractList** (p. 173) should be used in preference to this class.

This class is the opposite of the **AbstractList** (p. 173) class in the sense that it implements the "random access" methods (`get(int index)`, `set(int index, E element)`, `add(int index, E element)` and `remove(int index)`) on top of the list's list iterator, instead of the other way around.

To implement a list the programmer needs only to extend this class and provide implementations for the `listIterator` and `size` methods. For an unmodifiable list, the programmer need only implement the list iterator's `hasNext`, `next`, `hasPrevious`, `previous` and `index` methods.

For a modifiable list the programmer should additionally implement the list iterator's `set` method. For a variable-size list the programmer should additionally implement the list iterator's `remove` and `add` methods.

The programmer should generally provide a void (no argument) and collection constructor, as per the recommendation in the **Collection** (p. 1216) interface specification.

### Since

1.0

## 6.5.2 Constructor & Destructor Documentation

6.5.2.1 `template<typename E> virtual decaf::util::AbstractSequentialList< E >::~~AbstractSequentialList( ) [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractSequentialList.h`

## 6.6 decaf::util::AbstractSet< E > Class Template Reference

This class provides a skeletal implementation of the **Set** (p. 3538) interface to minimize the effort required to implement this interface.

```
#include <src/main/decaf/util/AbstractSet.h>
```

Inheritance diagram for `decaf::util::AbstractSet< E >`:

### Public Member Functions

- `virtual ~AbstractSet( )`
- `virtual bool removeAll(const Collection< E > &collection) throw (lang::exceptions::UnsupportedOperationException lang::exceptions::IllegalArgumentException)`

*Removes from this set all of its elements that are contained in the specified collection (optional operation).*

### 6.6.1 Detailed Description

`template<typename E> class decaf::util::AbstractSet< E >`

This class provides a skeletal implementation of the **Set** (p. 3538) interface to minimize the effort required to implement this interface. The process of implementing a set by extending this class is identical to that of implementing a **Collection** (p. 1216) by extending **AbstractCollection** (p. 159), except that all of the methods and constructors in subclasses of this class must obey the additional constraints imposed by the **Set** (p. 3538) interface (for instance, the add method must not permit addition of multiple instances of an object to a set).

Since

1.0

## 6.6.2 Constructor & Destructor Documentation

6.6.2.1 `template<typename E> virtual decaf::util::AbstractSet< E >::~~AbstractSet ( ) [inline, virtual]`

## 6.6.3 Member Function Documentation

6.6.3.1 `template<typename E> virtual bool decaf::util::AbstractSet< E >::removeAll ( const Collection< E > & collection ) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException ) [inline, virtual]`

Removes from this set all of its elements that are contained in the specified collection (optional operation).

If the specified collection is also a set, this operation effectively modifies this set so that its value is the asymmetric set difference of the two sets.

This implementation determines which is the smaller of this set and the specified collection, by invoking the size method on each. If this set has fewer elements, then the implementation iterates over this set, checking each element returned by the iterator in turn to see if it is contained in the specified collection. If it is so contained, it is removed from this set with the iterator's remove method. If the specified collection has fewer elements, then the implementation iterates over the specified collection, removing from this set each element returned by the iterator, using this set's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method.

### Parameters

<i>collection</i>	- The <b>Collection</b> (p. 1216) whose elements are to be retained
-------------------	---

### Returns

true if the collection changed as a result of this call

### Exceptions

<i>UnsupportedOperationException</i>	
<i>IllegalArgumentException</i>	

Reimplemented from `decaf::util::AbstractCollection< E >` (p. 169).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractSet.h`

## 6.7 activemq::transport::AbstractTransportFactory Class Reference

Abstract implementation of the **TransportFactory** (p. 4003) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 4003) instances.

```
#include <src/main/activemq/transport/AbstractTransportFactory.h>
```

Inheritance diagram for activemq::transport::AbstractTransportFactory:

### Public Member Functions

- virtual `~AbstractTransportFactory()`

### Protected Member Functions

- virtual `Pointer< wireformat::WireFormat > createWireFormat (const decaf::util::Properties &properties) throw ( decaf::lang::exceptions::NoSuchElementException )`

*Creates the WireFormat that is configured for this **Transport** (p. 3996) and returns it.*

#### 6.7.1 Detailed Description

Abstract implementation of the **TransportFactory** (p. 4003) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 4003) instances.

Since

3.0

#### 6.7.2 Constructor & Destructor Documentation

6.7.2.1 virtual `activemq::transport::AbstractTransportFactory::~AbstractTransportFactory()`   
 [inline, virtual]

#### 6.7.3 Member Function Documentation

6.7.3.1 virtual `Pointer<wireformat::WireFormat> activemq::transport::AbstractTransportFactory::createWireFormat ( const decaf::util::Properties & properties ) throw ( decaf::lang::exceptions::NoSuchElementException )` [protected, virtual]

Creates the WireFormat that is configured for this **Transport** (p. 3996) and returns it.

The default WireFormat is Openwire.

**Parameters**

<i>properties</i>	The properties that were configured on the URI.
-------------------	---

**Returns**

a pointer to a WireFormat instance that the caller then owns.

**Exceptions**

<i>NoSuchElementException</i>	if the configured WireFormat is not found.
-------------------------------	--

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**AbstractTransportFactory.h**

## 6.8 activemq::core::ActiveMQAckHandler Class Reference

Interface class that is used to give CMS Messages an interface to Ack themselves with.

```
#include <src/main/activemq/core/ActiveMQAckHandler.h>
```

**Public Member Functions**

- virtual **~ActiveMQAckHandler** ()
- virtual void **acknowledgeMessage** (const **commands::Message** \*message)=0  
throw ( cms::CMSException )

*Method called to acknowledge the message passed.*

### 6.8.1 Detailed Description

Interface class that is used to give CMS Messages an interface to Ack themselves with.

**Since**

2.0



## 6.8.2 Constructor & Destructor Documentation

6.8.2.1 virtual `activemq::core::ActiveMQAckHandler::~~ActiveMQAckHandler ( )`  
`[inline, virtual]`

## 6.8.3 Member Function Documentation

6.8.3.1 virtual void `activemq::core::ActiveMQAckHandler::acknowledgeMessage ( const commands::Message * message ) throw ( cms::CMSException )` `[pure virtual]`

Method called to acknowledge the message passed.

### Parameters

<i>message</i>	Message to Acknowledge
----------------	------------------------

### Exceptions

<i>CMSException</i>	
---------------------	--

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQAckHandler.h`

## 6.9 activemq::commands::ActiveMQBlobMessage Class Reference

```
#include <src/main/activemq/commands/ActiveMQBlobMessage.h>
```

Inheritance diagram for `activemq::commands::ActiveMQBlobMessage`:

### Public Member Functions

- **ActiveMQBlobMessage ( )**
- virtual **~ActiveMQBlobMessage ( )**
- virtual unsigned char **getDataStructureType ( )** const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ActiveMQBlobMessage \* cloneDataStructure ( )** const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure (const DataStructure \*src)**  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** \*value) const

*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*

- virtual **cms::Message** \* **clone** () const

*Clone this message exactly, returns a new instance that the caller is required to delete.*

- std::string **getRemoteBlobUrl** () const

*Get the Remote URL of the Blob.*

- void **setRemoteBlobUrl** (const std::string &remoteURL)

*Set the Remote URL of the Blob.*

- std::string **getMimeType** () const

*Get the Mime Type of the Blob.*

- void **setMimeType** (const std::string &mimeType)

*Set the Mime Type of the Blob.*

- std::string **getName** () const

*Gets the Name of the Blob.*

- void **setName** (const std::string &name)

*Sets the Name of the Blob.*

- bool **isDeletedByBroker** () const

*Gets if this Blob is deleted by the Broker.*

- void **setDeletedByBroker** (bool value)

*Sets the Deleted By Broker flag.*

## Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQBLOBMESSAGE** = 29
- static const std::string **BINARY\_MIME\_TYPE**

## 6.9.1 Constructor & Destructor Documentation

6.9.1.1 `activemq::commands::ActiveMQBlobMessage::ActiveMQBlobMessage ( )`

6.9.1.2 `virtual activemq::commands::ActiveMQBlobMessage::~~ActiveMQBlobMessage ( )`  
[inline, virtual]

## 6.9.2 Member Function Documentation

6.9.2.1 `virtual cms::Message* activemq::commands::ActiveMQBlobMessage::clone ( )`  
`const` [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

### Returns

new copy of this message

Implements `cms::Message` (p. 2620).

6.9.2.2 `virtual ActiveMQBlobMessage* activemq::commands::ActiveMQBlobMessage::cloneDataStructure ( ) const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2601).

6.9.2.3 `virtual void activemq::commands::ActiveMQBlobMessage::copyDataStructure ( const DataStructure* src )` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

### Returns

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2602).

6.9.2.4 `virtual bool activemq::commands::ActiveMQBlobMessage::equals ( const DataStructure* value ) const` [virtual]

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns**

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 423).

**6.9.2.5** `virtual unsigned char activemq::commands::ActiveMQBlobMessage::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns**

new **DataStructure** (p. 1713) type copy.

Reimplemented from `activemq::commands::Message` (p. 2604).

**6.9.2.6** `std::string activemq::commands::ActiveMQBlobMessage::getMimeType ( ) const [inline]`

Get the Mime Type of the Blob.

**Returns**

string holding the MIME Type.

**6.9.2.7** `std::string activemq::commands::ActiveMQBlobMessage::getName ( ) const [inline]`

Gets the Name of the Blob.

**Returns**

string name of the Blob.

**6.9.2.8** `std::string activemq::commands::ActiveMQBlobMessage::getRemoteBlobUrl ( ) const [inline]`

Get the Remote URL of the Blob.

**Returns**

string from of the Remote Blob URL.

**6.9.2.9** `bool activemq::commands::ActiveMQBlobMessage::isDeletedByBroker ( ) const`  
[inline]

Gets if this Blob is deleted by the Broker.

#### Returns

true if the Blob is deleted by the Broker.

**6.9.2.10** `void activemq::commands::ActiveMQBlobMessage::setDeletedByBroker ( bool value )`  
[inline]

Sets the Deleted By Broker flag.

#### Parameters

<i>value</i>	- set the Delete by broker flag to value.
--------------	---

**6.9.2.11** `void activemq::commands::ActiveMQBlobMessage::setMimeType ( const std::string & mimeType )` [inline]

Set the Mime Type of the Blob.

#### Parameters

<i>mimeType</i>	- String holding the MIME Type.
-----------------	---------------------------------

**6.9.2.12** `void activemq::commands::ActiveMQBlobMessage::setName ( const std::string & name )` [inline]

Sets the Name of the Blob.

#### Parameters

<i>name</i>	- Name of the Blob.
-------------	---------------------

**6.9.2.13** `void activemq::commands::ActiveMQBlobMessage::setRemoteBlobUrl ( const std::string & remoteURL )` [inline]

Set the Remote URL of the Blob.

#### Parameters

<i>remoteURL</i>	- String form of the Remote URL.
------------------	----------------------------------

**6.9.2.14** `virtual std::string activemq::commands::ActiveMQBlobMessage::toString ( ) const`  
`[virtual]`

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 2611).

### 6.9.3 Field Documentation

**6.9.3.1** `const std::string activemq::commands::ActiveMQBlobMessage::BINARY_MIME_TYPE` `[static]`

**6.9.3.2** `const unsigned char activemq::commands::ActiveMQBlobMessage::ID_ACTIVEMQBLOBMESSAGE = 29` `[static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQBlobMessage.h`

## 6.10 **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 189).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMes
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller**:

### Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*

## 6.10 ac-

**ativemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller**

### Class Reference

191

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.10.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 189).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.10.2 Constructor & Destructor Documentation

6.10.2.1 **ativemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller**  
( ) [inline]

6.10.2.2 **virtual ativemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller**  
( ) [inline, virtual]

### 6.10.3 Member Function Documentation

6.10.3.1 **virtual commands::DataStructure\* ativemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.10.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.10.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2786).

**6.10.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---



## 6.10 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller

### Class Reference 193

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2786).

```
6.10.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2787).

```
6.10.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink

<i>bs</i>	- BooleanStream stream used to pack bits from the wire.
-----------	---

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2788).

```
6.10.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2788).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h

## 6.11 activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 194).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMes
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller**:

- **ActiveMQBlobMessageMarshaller ()**
- virtual **~ActiveMQBlobMessageMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.11.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 194).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.11.2 Constructor & Destructor Documentation

6.11.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller ( ) [inline]`

6.11.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller ( ) [inline, virtual]`

### 6.11.3 Member Function Documentation

6.11.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.11.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.11.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## 6.11 activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller

### Class Reference

### Exceptions

197

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2799).

**6.11.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2800).

**6.11.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2800).

**6.11.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2801).

**6.11.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2802).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h`

## Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 198).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBlobMessageMarsh
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller`:

## Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.12.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 198).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.12.2 Constructor & Destructor Documentation

6.12.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller ( ) [inline]`

6.12.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller ( ) [inline, virtual]`

### 6.12.3 Member Function Documentation

6.12.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.12.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.12.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters



## 6.12 ac-

### activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller Class Reference 201

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2795).

**6.12.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2795).

**6.12.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2796).

**6.12.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2797).

**6.12.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### 6.13 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller

**Class Reference** 203

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2797).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBlobMessageMarshaller.h

## 6.13 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 202).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller**:

#### Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.13.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 202).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.13.2 Constructor & Destructor Documentation

6.13.2.1 **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller**  
( ) [inline]

6.13.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller**  
( ) [inline, virtual]

### 6.13.3 Member Function Documentation

6.13.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.13.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

### 6.13 ac-

**activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller**

**Class Reference**

**205**

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.13.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2790).

**6.13.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2791).

**6.13.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2791).

**6.13.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2792).

## 6.14 ac-

**activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller**

### Class Reference

207

6.13.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2793).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h`

## 6.14 **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 206).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarsh
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller**:

### Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.14.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 206).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.14.2 Constructor & Destructor Documentation

6.14.2.1 **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller**  
( ) [inline]

6.14.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller**  
( ) [inline, virtual]

### 6.14.3 Member Function Documentation

6.14.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.



new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.14.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::getDataStructureType ( ) const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.14.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2781).

**6.14.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2782).

```
6.14.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2783).

```
6.14.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink

## 6.15 activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller Class Reference 211

<i>bs</i>	- BooleanStream stream used to pack bits from the wire.
-----------	---

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2783).

**6.14.3.7** virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarshaller.h

## 6.15 activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 211).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBlobMessageMarsh
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller:

## Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**)  
**throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) **throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) **throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) **throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) **throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.15.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 211).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

6.15.2.1 **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller**  
( ) [inline]

6.15.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller**  
( ) [inline, virtual]

### 6.15.3 Member Function Documentation

6.15.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.15.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.15.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2804).

**6.15.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2804).

**6.15.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.15 ac-

**activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller**

### Class Reference

215

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2805).

**6.15.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2806).

**6.15.3.7** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2806).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBlobMessageMarshaller.h

## 6.16 activemq::commands::ActiveMQBytesMessage Class Reference

```
#include <src/main/activemq/commands/ActiveMQBytesMessage.h>
```

Inheritance diagram for activemq::commands::ActiveMQBytesMessage:

### Public Member Functions

- **ActiveMQBytesMessage** ()
- virtual **~ActiveMQBytesMessage** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ActiveMQBytesMessage \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual **cms::BytesMessage \* clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*
- virtual void **clearBody** () throw ( cms::CMSException )  
*Clears out the body of the message.*
- virtual void **onSend** ()  
*Store the Data that was written to the stream before a send.*
- virtual void **setBodyBytes** (const unsigned char \*buffer, int numBytes) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*sets the bytes given to the message body.*
- virtual unsigned char \* **getBodyBytes** () const throw ( cms::MessageNotReadableException, cms::CMSException )



*Gets the bytes that are contained in this message, user should copy this data into a user allocated buffer.*

- virtual int **getBodyLength** () const throw ( cms::MessageNotReadableException, cms::CMSEException )

*Returns the number of bytes contained in the body of this message.*

- virtual void **reset** () throw ( cms::MessageFormatException, cms::CMSEException )

*Puts the message body in read-only mode and repositions the stream of bytes to the beginning.*

- virtual bool **readBoolean** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a Boolean from the Bytes message stream.*

- virtual void **writeBoolean** (bool value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a boolean to the bytes message stream as a 1-byte value.*

- virtual unsigned char **readByte** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a Byte from the Bytes message stream.*

- virtual void **writeByte** (unsigned char value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a byte to the bytes message stream as a 1-byte value.*

- virtual int **readBytes** (std::vector< unsigned char > &value) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a byte array from the bytes message stream.*

- virtual void **writeBytes** (const std::vector< unsigned char > &value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.*

- virtual int **readBytes** (unsigned char \*buffer, int length) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a portion of the bytes message stream.*

- virtual void **writeBytes** (const unsigned char \*value, int offset, int length) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a portion of a byte array to the bytes message stream.*

- virtual char **readChar** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a Char from the Bytes message stream.*

- virtual void **writeChar** (char value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a char to the bytes message stream as a 1-byte value.*

- virtual float **readFloat** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 32 bit float from the Bytes message stream.*

- virtual void **writeFloat** (float value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a float to the bytes message stream as a 4 byte value.*

- virtual double **readDouble** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 64 bit double from the Bytes message stream.*

- virtual void **writeDouble** (double value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a double to the bytes message stream as a 8 byte value.*

- virtual short **readShort** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 16 bit signed short from the Bytes message stream.*

- virtual void **writeShort** (short value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a signed short to the bytes message stream as a 2 byte value.*

- virtual unsigned short **readUnsignedShort** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 16 bit unsigned short from the Bytes message stream.*

- virtual void **writeUnsignedShort** (unsigned short value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a unsigned short to the bytes message stream as a 2 byte value.*

- virtual int **readInt** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 32 bit signed integer from the Bytes message stream.*

- virtual void **writeInt** (int value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a signed int to the bytes message stream as a 4 byte value.*

- virtual long long **readLong** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 64 bit long from the Bytes message stream.*
- virtual void **writeLong** (long long value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a long long to the bytes message stream as a 8 byte value.*
- virtual std::string **readString** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads an ASCII String from the Bytes message stream.*
- virtual void **writeString** (const std::string &value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes an ASCII String to the Bytes message stream.*
- virtual std::string **readUTF** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads an UTF String from the BytesMessage stream.*
- virtual void **writeUTF** (const std::string &value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes an UTF String to the BytesMessage stream.*

### Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQBYTESMESSAGE** = 24

### 6.16.1 Constructor & Destructor Documentation

6.16.1.1 **activemq::commands::ActiveMQBytesMessage::ActiveMQBytesMessage ( )**

6.16.1.2 **virtual activemq::commands::ActiveMQBytesMessage::~~ActiveMQBytesMessage ( )**  
[virtual]

### 6.16.2 Member Function Documentation

6.16.2.1 **virtual void activemq::commands::ActiveMQBytesMessage::clearBody ( ) throw ( cms::CMSEException )** [virtual]

Clears out the body of the message.

This does not clear the headers or properties.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 422).

**6.16.2.2** `virtual cms::BytesMessage* activemq::commands::ActiveMQBytesMessage::clone ( ) const [inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

#### Returns

new copy of this message

Implements **cms::BytesMessage** (p. 1083).

**6.16.2.3** `virtual ActiveMQBytesMessage* activemq::commands::ActiveMQBytesMessage::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 2601).

**6.16.2.4** `virtual void activemq::commands::ActiveMQBytesMessage::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 2602).

**6.16.2.5** `virtual bool activemq::commands::ActiveMQBytesMessage::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Parameters

<i>value</i>	The <b>Command</b> (p. 1227) to compare to this one.
--------------	--

**Returns**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 423).

**6.16.2.6** `virtual unsigned char* activemq::commands::ActiveMQBytesMessage::getBodyBytes ( ) const throw ( cms::MessageNotReadableException, cms::CMSEException )`  
[virtual]

Gets the bytes that are contained in this message, user should copy this data into a user allocated buffer.

Call `getBodyLength` to determine the number of bytes to expect.

**Returns**

const pointer to a byte buffer

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
<i>MessageNotReadableException</i>	- If the message is in Write Only Mode.

Implements **cms::BytesMessage** (p. 1083).

**6.16.2.7** `virtual int activemq::commands::ActiveMQBytesMessage::getBodyLength ( ) const throw ( cms::MessageNotReadableException, cms::CMSEException )`  
[virtual]

Returns the number of bytes contained in the body of this message.

**Returns**

number of bytes.

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
<i>MessageNotReadableException</i>	- If the message is in Write Only Mode.

Implements **cms::BytesMessage** (p. 1084).

**6.16.2.8** `virtual unsigned char activemq::commands::ActiveMQBytesMessage::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1713) type copy.

Reimplemented from **activemq::commands::Message** (p. 2604).

**6.16.2.9** `virtual void activemq::commands::ActiveMQBytesMessage::onSend ( ) [virtual]`

Store the Data that was written to the stream before a send.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 431).

**6.16.2.10** `virtual bool activemq::commands::ActiveMQBytesMessage::readBoolean ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [virtual]`

Reads a Boolean from the Bytes message stream.

#### Returns

boolean value from stream

#### Exceptions

<i>CMSEException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1084).

**6.16.2.11** `virtual unsigned char activemq::commands::ActiveMQBytesMessage::readByte ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [virtual]`

Reads a Byte from the Bytes message stream.

**Returns**

unsigned char value from stream

**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1085).

**6.16.2.12** `virtual int activemq::commands::ActiveMQBytesMessage::readBytes ( std::vector< unsigned char > & value ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException )`  
[virtual]

Reads a byte array from the bytes message stream.

If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

**Parameters**

<i>value</i>	buffer to place data in
--------------	-------------------------

**Returns**

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1085).

**6.16.2.13** `virtual int activemq::commands::ActiveMQBytesMessage::readBytes ( unsigned char * buffer, int length ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException )`  
`[virtual]`

Reads a portion of the bytes message stream.

If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an `IndexOutOfBoundsException` is thrown. No bytes will be read from the stream for this exception case.

#### Parameters

<i>buffer</i>	the buffer into which the data is read
<i>length</i>	the number of bytes to read; must be less than or equal to value.length

#### Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements `cms::BytesMessage` (p. 1086).

**6.16.2.14** `virtual char activemq::commands::ActiveMQBytesMessage::readChar ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException )` `[virtual]`

Reads a Char from the Bytes message stream.

#### Returns

char value from stream



**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1087).

**6.16.2.15** `virtual double activemq::commands::ActiveMQBytesMessage::readDouble ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [virtual]`

Reads a 64 bit double from the Bytes message stream.

**Returns**

double value from stream

**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1087).

**6.16.2.16** `virtual float activemq::commands::ActiveMQBytesMessage::readFloat ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [virtual]`

Reads a 32 bit float from the Bytes message stream.

**Returns**

double value from stream

**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1088).

**6.16.2.17** `virtual int activemq::commands::ActiveMQBytesMessage::readInt ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [virtual]`

Reads a 32 bit signed integer from the Bytes message stream.

#### Returns

int value from stream

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1088).

**6.16.2.18** `virtual long long activemq::commands::ActiveMQBytesMessage::readLong ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [virtual]`

Reads a 64 bit long from the Bytes message stream.

#### Returns

long long value from stream

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1089).

**6.16.2.19** virtual short activemq::commands::ActiveMQBytesMessage::readShort  
 ( ) const throw ( cms::MessageEOFException,  
 cms::MessageNotReadableException, cms::CMSEException )  
 [virtual]

Reads a 16 bit signed short from the Bytes message stream.

#### Returns

short value from stream

#### Exceptions

<i>CMSEException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1089).

**6.16.2.20** virtual std::string activemq::commands::ActiveMQBytesMessage::readString  
 ( ) const throw ( cms::MessageEOFException,  
 cms::MessageNotReadableException, cms::CMSEException )  
 [virtual]

Reads an ASCII String from the Bytes message stream.

#### Returns

String from stream

#### Exceptions

<i>CMSEException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1090).

**6.16.2.21** `virtual unsigned short activemq::commands::ActiveMQBytesMessage::readUnsignedShort ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [virtual]`

Reads a 16 bit unsigned short from the Bytes message stream.

#### Returns

unsigned short value from stream

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1090).

**6.16.2.22** `virtual std::string activemq::commands::ActiveMQBytesMessage::readUTF ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [virtual]`

Reads an UTF String from the BytesMessage stream.

#### Returns

String from stream

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1091).

**6.16.2.23** `virtual void activemq::commands::ActiveMQBytesMessage::reset ( ) throw ( cms::MessageFormatException, cms::CMSException ) [virtual]`

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

**Exceptions**

<i>CMSException</i>	- If the provider fails to perform the reset operation.
<i>MessageFormatException</i>	- If the <b>Message</b> (p. 2596) has an invalid format.

Implements **cms::BytesMessage** (p. 1091).

**6.16.2.24** `virtual void activemq::commands::ActiveMQBytesMessage::setBodyBytes ( const unsigned char * buffer, int numBytes ) throw ( cms::MessageNotWriteableException, cms::CMSException )`  
[virtual]

sets the bytes given to the message body.

**Parameters**

<i>buffer</i>	Byte Buffer to copy
<i>numBytes</i>	Number of bytes in Buffer to copy

**Exceptions**

<i>CMSException</i>	- If an internal error occurs.
<i>MessageNotWriteableException</i>	- if in Read Only Mode.

Implements **cms::BytesMessage** (p. 1091).

**6.16.2.25** `virtual std::string activemq::commands::ActiveMQBytesMessage::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

**Returns**

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 2611).

**6.16.2.26** `virtual void activemq::commands::ActiveMQBytesMessage::writeBoolean ( bool value ) throw ( cms::MessageNotWriteableException, cms::CMSException )`  
[virtual]

Writes a boolean to the bytes message stream as a 1-byte value.

The value true is written as the value (byte)1; the value false is written as the value (byte)0.

**Parameters**

<i>value</i>	boolean to write to the stream
--------------	--------------------------------

**Exceptions**

<i>CMSEException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1092).

**6.16.2.27** `virtual void activemq::commands::ActiveMQBytesMessage::writeByte ( unsigned char value ) throw ( cms::MessageNotWriteableException, cms::CMSEException )` [virtual]

Writes a byte to the bytes message stream as a 1-byte value.

**Parameters**

<i>value</i>	byte to write to the stream
--------------	-----------------------------

**Exceptions**

<i>CMSEException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1092).

**6.16.2.28** `virtual void activemq::commands::ActiveMQBytesMessage::writeBytes ( const unsigned char * value, int offset, int length ) throw ( cms::MessageNotWriteableException, cms::CMSEException )` [virtual]

Writes a portion of a byte array to the bytes message stream.

size as the number of bytes to write.

**Parameters**

<i>value</i>	bytes to write to the stream
<i>offset</i>	the initial offset within the byte array
<i>length</i>	the number of bytes to use

**Exceptions**

<i>CMSEException</i>	- if the CMS provider fails to write the message due to some internal error.
----------------------	--

<i>MessageNotWriteableException</i>	- if the message is in read-only mode.
-------------------------------------	--

Implements **cms::BytesMessage** (p. 1093).

**6.16.2.29** `virtual void activemq::commands::ActiveMQBytesMessage::writeBytes  
( const std::vector< unsigned char > & value ) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
[virtual]`

Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

#### Parameters

<i>value</i>	bytes to write to the stream
--------------	------------------------------

#### Exceptions

<i>CMSEException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1093).

**6.16.2.30** `virtual void activemq::commands::ActiveMQBytesMessage::writeChar ( char value )  
throw ( cms::MessageNotWriteableException, cms::CMSEException )  
[virtual]`

Writes a char to the bytes message stream as a 1-byte value.

#### Parameters

<i>value</i>	char to write to the stream
--------------	-----------------------------

#### Exceptions

<i>CMSEException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1094).

**6.16.2.31** `virtual void activemq::commands::ActiveMQBytesMessage::writeDouble  
( double value ) throw ( cms::MessageNotWriteableException,  
cms::CMSException ) [virtual]`

Writes a double to the bytes message stream as a 8 byte value.

#### Parameters

<i>value</i>	double to write to the stream
--------------	-------------------------------

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1094).

**6.16.2.32** `virtual void activemq::commands::ActiveMQBytesMessage::writeFloat ( float value )  
throw ( cms::MessageNotWriteableException, cms::CMSException )  
[virtual]`

Writes a float to the bytes message stream as a 4 byte value.

#### Parameters

<i>value</i>	float to write to the stream
--------------	------------------------------

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1094).

**6.16.2.33** `virtual void activemq::commands::ActiveMQBytesMessage::writeInt ( int value )  
throw ( cms::MessageNotWriteableException, cms::CMSException )  
[virtual]`

Writes a signed int to the bytes message stream as a 4 byte value.

#### Parameters

<i>value</i>	signed int to write to the stream
--------------	-----------------------------------



**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1095).

**6.16.2.34** `virtual void activemq::commands::ActiveMQBytesMessage::writeLong ( long long value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Writes a long long to the bytes message stream as a 8 byte value.

**Parameters**

<i>value</i>	signed long long to write to the stream
--------------	---

**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1095).

**6.16.2.35** `virtual void activemq::commands::ActiveMQBytesMessage::writeShort ( short value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Writes a signed short to the bytes message stream as a 2 byte value.

**Parameters**

<i>value</i>	signed short to write to the stream
--------------	-------------------------------------

**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1096).

**6.16.2.36** `virtual void activemq::commands::ActiveMQBytesMessage::writeString ( const std::string & value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Writes an ASCII String to the Bytes message stream.

#### Parameters

<i>value</i>	String to write to the stream
--------------	-------------------------------

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1096).

**6.16.2.37** `virtual void activemq::commands::ActiveMQBytesMessage::writeUnsignedShort ( unsigned short value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Writes a unsigned short to the bytes message stream as a 2 byte value.

#### Parameters

<i>value</i>	unsigned short to write to the stream
--------------	---------------------------------------

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1096).

**6.16.2.38** `virtual void activemq::commands::ActiveMQBytesMessage::writeUTF ( const std::string & value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Writes an UTF String to the BytesMessage stream.

#### Parameters

<i>value</i>	String to write to the stream
--------------	-------------------------------

## 6.17 activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller

### Class Reference

#### Exceptions

235

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1097).

### 6.16.3 Field Documentation

6.16.3.1 **const unsigned char activemq::commands::ActiveMQBytesMessage::ID\_ - ACTIVEMQBYTESMESSAGE = 24** [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/ActiveMQBytesMessage.h

## 6.17 activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 235).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller**:

### Public Member Functions

- **ActiveMQBytesMessageMarshaller ()**
- virtual **~ActiveMQBytesMessageMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.17.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 235).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.17.2 Constructor & Destructor Documentation

- 6.17.2.1 **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller**  
 ( ) [inline]
- 6.17.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller**  
 ( ) [inline, virtual]

### 6.17.3 Member Function Documentation

- 6.17.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

## 6.17 ac-

**activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller**

### Class Reference

237

**6.17.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaller.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.17.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2786).

**6.17.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2786).

```
6.17.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2787).

```
6.17.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2788).

## 6.18 ac-

ativemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller

### Class Reference

239

6.17.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

wireFormat	- describes the wire format of the broker.
dataStructure	- Object to be un-marshaled.
dataIn	- BinaryReader that provides that data.
bs	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

IOException	if an error occurs during the unmarshal.
-------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2788).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h

## 6.18 activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 239).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBytesMessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller**:

### Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual **~ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.18.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 239).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.18.2 Constructor & Destructor Documentation

6.18.2.1 **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller**  
( ) [inline]

6.18.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller**  
( ) [inline, virtual]

### 6.18.3 Member Function Documentation

6.18.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.



## 6.18 activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller

### Class Reference

---

### Returns

241

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.18.3.2** virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.18.3.3** virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::looseMarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2799).

**6.18.3.4** virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::looseUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2800).

```
6.18.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2800).

```
6.18.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink

## 6.19 activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller

### Class Reference

243

<i>bs</i>	- BooleanStream stream used to pack bits from the wire.
-----------	---

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2801).

**6.18.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2802).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBytesMessageMarshaller.h

## 6.19 activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 243).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller:

## Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual **~ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.19.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 243).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.19.2.1 **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller**  
( ) [inline]

6.19.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller**  
( ) [inline, virtual]

### 6.19.3 Member Function Documentation

6.19.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.19.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.19.3.3 **virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2795).

**6.19.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2795).

**6.19.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.19 ac-

**activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller**

### Class Reference

247

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2796).

**6.19.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2797).

**6.19.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2797).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h

## 6.20 activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 247).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBytesMe
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`:

### Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual **~ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*



Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 247).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

## 6.20.2 Constructor & Destructor Documentation

6.20.2.1 **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller**  
 ( ) [inline]

6.20.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller**  
 ( ) [inline, virtual]

## 6.20.3 Member Function Documentation

6.20.3.1 **virtual commands::DataStructure\* ac-**  
**tivemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.20.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.20.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::looseMarshal**  
 ( **OpenWireFormat \* wireFormat**, **commands::DataStructure**  
 \* **dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2781).

```
6.20.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataInputStream * dataIn ) throw (
 decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2782).

```
6.20.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
 dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2783).

**6.20.3.6** virtual void **ativemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::tightMarshal2**  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
 \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*,  
**utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2783).

**6.20.3.7** virtual void **ativemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::tightUnmarshal**  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \*  
*bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBytesMessageMarshaller.h

## 6.21 **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 252).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBytesMe
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller**:

**Public Member Functions**

- **ActiveMQBytesMessageMarshaller** ()
- virtual **~ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

## 6.21 ac-

**ativemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller**

### Class Reference

253

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.21.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 252).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.21.2 Constructor & Destructor Documentation

6.21.2.1 **ativemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller**  
( ) [inline]

6.21.2.2 **virtual ativemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller**  
( ) [inline, virtual]

### 6.21.3 Member Function Documentation

6.21.3.1 **virtual commands::DataStructure\* ac-**  
**ativemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **ativemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.21.3.2 **virtual unsigned char ativemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

```
6.21.3.3 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2804).

```
6.21.3.4 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2804).

## 6.21 ac-

**activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller**

### Class Reference

255

6.21.3.5 **virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::tightMarshal1**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **utils::BooleanStream** \* *bs* ) **throw ( decaf::io::IOException )**  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller**  
(p. 2805).

6.21.3.6 **virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::tightMarshal2**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
\* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*,  
**utils::BooleanStream** \* *bs* ) **throw ( decaf::io::IOException )** [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller**  
(p. 2806).

```
6.21.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2806).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBytesMessageMarshaller.h

## 6.22 activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 256).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMe
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller**:

#### Public Member Functions

- **ActiveMQBytesMessageMarshaller ()**
- virtual **~ActiveMQBytesMessageMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*



## 6.22 ac-

**ativemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller**

### Class Reference

257

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

## 6.22.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 256).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.22.2 Constructor & Destructor Documentation

6.22.2.1 **ativemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller**  
( ) [inline]

6.22.2.2 **virtual ativemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller**  
( ) [inline, virtual]

## 6.22.3 Member Function Documentation

6.22.3.1 **virtual commands::DataStructure\* ativemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.22.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.22.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2790).

**6.22.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

## 6.22 activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller

### Class Reference 259

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2791).

**6.22.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2791).

**6.22.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink

<i>bs</i>	- BooleanStream stream used to pack bits from the wire.
-----------	---

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2792).

```
6.22.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2793).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h

## 6.23 activemq::core::ActiveMQConnection Class Reference

Concrete connection used for all connectors to the ActiveMQ broker.

```
#include <src/main/activemq/core/ActiveMQConnection.h>
```

Inheritance diagram for **activemq::core::ActiveMQConnection**:

### Public Member Functions

- **ActiveMQConnection** (const **Pointer**< **transport::Transport** > &transport,

const **Pointer**< **decaf::util::Properties** > &properties)

*Constructor.*

- virtual ~**ActiveMQConnection** ()
- virtual void **removeSession** (**ActiveMQSession** \*session) throw ( cms::CMSException )  
*Removes the session resources for the given session instance.*
- virtual void **addProducer** (**ActiveMQProducer** \*producer) throw ( cms::CMSException )  
*Adds an active Producer to the Set of known producers.*
- virtual void **removeProducer** (const **Pointer**< **commands::ProducerId** > &producerId) throw ( cms::CMSException )  
*Removes an active Producer to the Set of known producers.*
- virtual void **addDispatcher** (const **Pointer**< **commands::ConsumerId** > &consumer, **Dispatcher** \*dispatcher) throw ( cms::CMSException )  
*Adds a dispatcher for a consumer.*
- virtual void **removeDispatcher** (const **Pointer**< **commands::ConsumerId** > &consumer) throw ( cms::CMSException )  
*Removes the dispatcher for a consumer.*
- virtual void **sendPullRequest** (const **commands::ConsumerInfo** \*consumer, long long timeout) throw ( exceptions::ActiveMQException )  
*If supported sends a message pull request to the service provider asking for the delivery of a new message.*
- bool **isClosed** () const  
*Checks if this connection has been closed.*
- bool **isStarted** () const  
*Check if this connection has been started.*
- bool **isTransportFailed** () const  
*Checks if the Connection's Transport has failed.*
- virtual void **destroyDestination** (const **commands::ActiveMQDestination** \*destination) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::UnsupportedOperationException, activemq::exceptions::ActiveMQException )  
*Requests that the Broker removes the given Destination.*
- virtual void **destroyDestination** (const **cms::Destination** \*destination) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::UnsupportedOperationException, activemq::exceptions::ActiveMQException )

*Requests that the Broker removes the given Destination.*

- virtual const **cms::ConnectionMetaData** \* **getMetaData** () const throw ( cms::CMSEException )

*Gets the metadata for this connection.*

- virtual **cms::Session** \* **createSession** () throw ( cms::CMSEException )

*Creates a new Session to work for this Connection.*

- virtual std::string **getClientID** () const

*Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the setClientID method.*

#### Returns

*Client Id String for this **Connection** (p. 1296).*

#### Exceptions

<b>CMSEException</b> (p. 1190)	<i>if the provider fails to return the client id or an internal error occurs.</i>
-----------------------------------	---

- virtual void **setClientID** (const std::string &clientID)

*Sets the client identifier for this connection.*

*The preferred way to assign a CMS client's client identifier is for it to be configured in a client-specific **ConnectionFactory** (p. 1361) object and transparently assigned to the **Connection** (p. 1296) object it creates.*

*If a client sets the client identifier explicitly, it must do so immediately after it creates the connection and before any other action on the connection is taken. After this point, setting the client identifier is a programming error that should throw an **IllegalStateException** (p. 2059).*

#### Parameters

clientID	<i>The unique client identifier to assign to the <b>Connection</b> (p. 1296).</i>
----------	---

#### Exceptions

<b>CMSEException</b> (p. 1190)	<i>if the provider fails to set the client id due to some internal error.</i>
<b>InvalidClientIDException</b>	<i>if the id given is somehow invalid or is a duplicate.</i>
<b>IllegalStateException</b> (p. 2059)	<i>if the client tries to set the id after a <b>Connection</b> (p. 1296) method has been called.</i>

- virtual **cms::Session** \* **createSession** (**cms::Session::AcknowledgeMode** ackMode) throw ( cms::CMSEException )

*Creates a new Session to work for this Connection using the specified acknowledgment mode.*

- virtual void **close** () throw ( cms::CMSEException )

*Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).*

- virtual void **start** () throw ( cms::CMSEException )  
*Starts or (restarts) a connections delivery of incoming messages.*
- virtual void **stop** () throw ( cms::CMSEException )  
*Stop the flow of incoming messages.*
- virtual cms::ExceptionListener \* **getExceptionListener** () const  
*Gets the registered Exception Listener for this connection.*
- virtual void **setExceptionListener** (cms::ExceptionListener \*listener)  
*Sets the registered Exception Listener for this connection.*
- void **setUsername** (const std::string &username)  
*Sets the username that should be used when creating a new connection.*
- const std::string & **getUsername** () const  
*Gets the username that this factory will use when creating a new connection instance.*
- void **setPassword** (const std::string &password)  
*Sets the password that should be used when creating a new connection.*
- const std::string & **getPassword** () const  
*Gets the password that this factory will use when creating a new connection instance.*
- void **setDefaultClientId** (const std::string &clientId)  
*Sets the Client Id.*
- void **setBrokerURL** (const std::string &brokerURL)  
*Sets the Broker URL that should be used when creating a new connection instance.*
- const std::string & **getBrokerURL** () const  
*Gets the Broker URL that this factory will use when creating a new connection instance.*
- void **setPrefetchPolicy** (PrefetchPolicy \*policy)  
*Sets the **PrefetchPolicy** (p. 3062) instance that this factory should use when it creates new Connection instances.*
- PrefetchPolicy \* **getPrefetchPolicy** () const  
*Gets the pointer to the current **PrefetchPolicy** (p. 3062) that is in use by this ConnectionFactory.*
- void **setRedeliveryPolicy** (RedeliveryPolicy \*policy)  
*Sets the **RedeliveryPolicy** (p. 3267) instance that this factory should use when it creates new Connection instances.*
- RedeliveryPolicy \* **getRedeliveryPolicy** () const

*Gets the pointer to the current **RedeliveryPolicy** (p. 3267) that is in use by this **ConnectionFactory**.*

- **bool isDispatchAsync () const**
- **void setDispatchAsync (bool value)**

*Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.*

- **bool isAlwaysSyncSend () const**

*Gets if the Connection should always send things Synchronously.*

- **void setAlwaysSyncSend (bool value)**

*Sets if the Connection should always send things Synchronously.*

- **bool isUseAsyncSend () const**

*Gets if the useAsyncSend option is set.*

- **void setUseAsyncSend (bool value)**

*Sets the useAsyncSend option.*

- **bool isUseCompression () const**

*Gets if the Connection is configured for Message body compression.*

- **void setUseCompression (bool value)**

*Sets whether Message body compression is enabled.*

- **unsigned int getSendTimeout () const**

*Gets the assigned send timeout for this Connector.*

- **void setSendTimeout (unsigned int timeout)**

*Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.*

- **unsigned int getCloseTimeout () const**

*Gets the assigned close timeout for this Connector.*

- **void setCloseTimeout (unsigned int timeout)**

*Sets the close timeout to use when sending the disconnect request.*

- **unsigned int getProducerWindowSize () const**

*Gets the configured producer window size for Producers that are created from this connector.*

- **void setProducerWindowSize (unsigned int windowSize)**

*Sets the size in Bytes of messages that a producer can send before it is blocked to await a **ProducerAck** from the broker that frees enough memory to allow another message to be sent.*



- long long **getNextSessionId** ()  
*Get the Next available Session Id.*
- long long **getNextTempDestinationId** ()  
*Get the Next Temporary Destination Id.*
- long long **getNextLocalTransactionId** ()  
*Get the Next Temporary Destination Id.*
- void **addTransportListener** (transport::TransportListener \*transportListener)  
  
*Adds a transport listener so that a client can be notified of events in the underlying transport, client's are always notified after the event has been processed by the Connection class.*
- void **removeTransportListener** (transport::TransportListener \*transportListener)  
  
*Removes a registered TransportListener from the Connection's set of Transport listeners, this listener will no longer receive any Transport related events.*
- virtual void **onCommand** (const Pointer< commands::Command > &command)  
  
*Event handler for the receipt of a non-response command from the transport.*
- virtual void **onException** (const decaf::lang::Exception &ex)  
  
*Event handler for an exception from a command transport.*
- virtual void **transportInterrupted** ()  
  
*The transport has suffered an interruption from which it hopes to recover.*
- virtual void **transportResumed** ()  
  
*The transport has resumed after an interruption.*
- const commands::ConnectionInfo & **getConnectionInfo** () const throw ( exceptions::ActiveMQException )  
  
*Gets the ConnectionInfo for this Object, if the Connection is not open than this method throws an exception.*
- const commands::ConnectionId & **getConnectionId** () const throw ( exceptions::ActiveMQException )  
  
*Gets the ConnectionId for this Object, if the Connection is not open than this method throws an exception.*
- transport::Transport & **getTransport** () const  
  
*Gets a reference to this object's Transport instance.*

- void **oneway** (**Pointer**< **commands::Command** > command) throw ( **activemq::exceptions::ActiveMQException** )  
*Sends a oneway message.*
- void **syncRequest** (**Pointer**< **commands::Command** > command, unsigned int timeout=0) throw ( **activemq::exceptions::ActiveMQException** )  
*Sends a synchronous request and returns the response from the broker.*
- virtual void **fire** (const **exceptions::ActiveMQException** &ex)  
*Notify the exception listener.*
- void **setTransportInterruptProcessingComplete** ()  
*Indicates that a Connection resource that is processing the transportInterrupted event has completed.*

### 6.23.1 Detailed Description

Concrete connection used for all connectors to the ActiveMQ broker.

Since

2.0

### 6.23.2 Constructor & Destructor Documentation

- 6.23.2.1 **activemq::core::ActiveMQConnection::ActiveMQConnection** ( const **Pointer**< **transport::Transport** > & *transport*, const **Pointer**< **decaf::util::Properties** > & *properties* )

Constructor.

#### Parameters

<i>transport</i>	The Transport requested for this connection to the Broker.
<i>properties</i>	The Properties that were defined for this connection

- 6.23.2.2 **virtual activemq::core::ActiveMQConnection::~~ActiveMQConnection** ( )  
[virtual]

### 6.23.3 Member Function Documentation

- 6.23.3.1 **virtual void activemq::core::ActiveMQConnection::addDispatcher** ( const **Pointer**< **commands::ConsumerId** > & *consumer*, **Dispatcher** \* *dispatcher* ) throw ( **cms::CMSException** ) [virtual]

Adds a dispatcher for a consumer.

**Parameters**

<i>consumer</i>	- The consumer for which to register a dispatcher.
<i>dispatcher</i>	- The dispatcher to handle incoming messages for the consumer.

**6.23.3.2** `virtual void activemq::core::ActiveMQConnection::addProducer ( ActiveMQProducer * producer ) throw ( cms::CMSException )`  
 [virtual]

Adds an active Producer to the Set of known producers.

**Parameters**

<i>producer</i>	- The Producer to add from the the known set.
-----------------	---

**6.23.3.3** `void activemq::core::ActiveMQConnection::addTransportListener ( transport::TransportListener * transportListener )`

Adds a transport listener so that a client can be notified of events in the underlying transport, client's are always notified after the event has been processed by the Connection class.

Client's should ensure that the registered listener does not block or take a long amount of time to execute in order to not degrade performance of this Connection.

**Parameters**

<i>transportListener</i>	The TransportListener instance to add to this Connection's set of listeners to notify of Transport events.
--------------------------	--

**6.23.3.4** `virtual void activemq::core::ActiveMQConnection::close ( ) throw ( cms::CMSException )` [virtual]

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

**Exceptions**

<i>CMSException</i>	
---------------------	--

Implements `cms::Connection` (p. 1298).

**6.23.3.5** `virtual cms::Session* activemq::core::ActiveMQConnection::createSession ( cms::Session::AcknowledgeMode ackMode ) throw ( cms::CMSException ) [virtual]`

Creates a new Session to work for this Connection using the specified acknowledgment mode.

#### Parameters

<i>ackMode</i>	the Acknowledgment Mode to use.
----------------	---------------------------------

#### Exceptions

<i>CMSException</i>	
---------------------	--

Implements **cms::Connection** (p. 1298).

**6.23.3.6** `virtual cms::Session* activemq::core::ActiveMQConnection::createSession ( ) throw ( cms::CMSException ) [virtual]`

Creates a new Session to work for this Connection.

#### Exceptions

<i>CMSException</i>	
---------------------	--

Implements **cms::Connection** (p. 1299).

**6.23.3.7** `virtual void activemq::core::ActiveMQConnection::destroyDestination ( const commands::ActiveMQDestination * destination ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::UnsupportedOperationException, activemq::exceptions::ActiveMQException ) [virtual]`

Requests that the Broker removes the given Destination.

Calling this method implies that the client is finished with the Destination and that no other messages will be sent or received for the given Destination. The Broker frees all resources it has associated with this Destination.

#### Parameters

<i>destination</i>	The Destination the Broker will be requested to remove.
--------------------	---

#### Exceptions

<i>NullPointerException</i>	If the passed Destination is Null
<i>IllegalStateException</i>	If the connection is closed.

<i>UnsupportedOperationException</i>	If the wire format in use does not support this operation.
<i>ActiveMQException</i>	If any other error occurs during the attempt to destroy the destination.

**6.23.3.8** virtual void activemq::core::ActiveMQConnection::destroyDestination ( const cms::Destination \* *destination* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::UnsupportedOperationException, activemq::exceptions::ActiveMQException ) [virtual]

Requests that the Broker removes the given Destination.

Calling this method implies that the client is finished with the Destination and that no other messages will be sent or received for the given Destination. The Broker frees all resources it has associated with this Destination.

#### Parameters

<i>destination</i>	The CMS Destination the Broker will be requested to remove.
--------------------	---

#### Exceptions

<i>NullPointerException</i>	If the passed Destination is Null
<i>IllegalStateException</i>	If the connection is closed.
<i>UnsupportedOperationException</i>	If the wire format in use does not support this operation.
<i>ActiveMQException</i>	If any other error occurs during the attempt to destroy the destination.

**6.23.3.9** virtual void activemq::core::ActiveMQConnection::fire ( const exceptions::ActiveMQException & *ex* ) [virtual]

Notify the exception listener.

#### Parameters

<i>ex</i>	the exception to fire
-----------	-----------------------

**6.23.3.10** const std::string& activemq::core::ActiveMQConnection::getBrokerURL ( ) const

Gets the Broker URL that this factory will use when creating a new connection instance.

**Returns**

brokerURL string

**6.23.3.11** `virtual std::string activemq::core::ActiveMQConnection::getClientID ( ) const`  
`[virtual]`

Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the setClientID method.

**Returns**

Client Id String for this **Connection** (p. 1296).

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	if the provider fails to return the client id or an internal error occurs.
---	--

Implements **cms::Connection** (p. 1299).

**6.23.3.12** `unsigned int activemq::core::ActiveMQConnection::getCloseTimeout ( ) const`

Gets the assigned close timeout for this Connector.

**Returns**

the close timeout configured in the connection uri

**6.23.3.13** `const commands::ConnectionId& activemq::core::ActiveMQConnection::getConnectionId ( ) const`  
`throw ( exceptions::ActiveMQException )`

Gets the ConnectionId for this Object, if the Connection is not open than this method throws an exception.

**6.23.3.14** `const commands::ConnectionInfo& activemq::core::ActiveMQConnection::getConnectionInfo ( )`  
`const throw ( exceptions::ActiveMQException )`

Gets the ConnectionInfo for this Object, if the Connection is not open than this method throws an exception.

**6.23.3.15** virtual cms::ExceptionListener\* activemq::core::ActiveMQConnection::getExceptionListener ( )  
const [virtual]

Gets the registered Exception Listener for this connection.

#### Returns

pointer to an exception listener or NULL

Implements cms::Connection (p. 1299).

**6.23.3.16** virtual const cms::ConnectionMetaData\* activemq::core::ActiveMQConnection::getMetaData ( ) const throw ( cms::CMSException ) [inline, virtual]

Gets the metadata for this connection.

#### Returns

the connection MetaData pointer ( caller does not own it ).

#### Exceptions

<i>CMSException</i>	if the provider fails to get the connection metadata for this connection.
---------------------	---

#### See also

ConnectionMetaData

#### Since

2.0

Implements cms::Connection (p. 1299).

**6.23.3.17** long long activemq::core::ActiveMQConnection::getNextLocalTransactionId ( )

Get the Next Temporary Destination Id.

#### Returns

the next id in the sequence.

**6.23.3.18** long long activemq::core::ActiveMQConnection::getNextSessionId ( )

Get the Next available Session Id.

**Returns**

the next id in the sequence.

**6.23.3.19** `long long activemq::core::ActiveMQConnection::getNextTempDestinationId ( )`

Get the Next Temporary Destination Id.

**Returns**

the next id in the sequence.

**6.23.3.20** `const std::string& activemq::core::ActiveMQConnection::getPassword ( ) const`

Gets the password that this factory will use when creating a new connection instance.

**Returns**

password string, "" for default credentials

**6.23.3.21** `PrefetchPolicy* activemq::core::ActiveMQConnection::getPrefetchPolicy ( ) const`

Gets the pointer to the current **PrefetchPolicy** (p. 3062) that is in use by this ConnectionFactory.

**Returns**

a pointer to this objects **PrefetchPolicy** (p. 3062).

**6.23.3.22** `unsigned int activemq::core::ActiveMQConnection::getProducerWindowSize ( ) const`

Gets the configured producer window size for Producers that are created from this connector.

This only applies if there is no send timeout and the producer is able to send asynchronously.

**Returns**

size in bytes of messages that this producer can produce before it must block and wait for ProducerAck messages to free resources.



**6.23.3.23 RedeliveryPolicy\* activemq::core::ActiveMQConnection::getRedeliveryPolicy ( ) const**

Gets the pointer to the current **RedeliveryPolicy** (p. 3267) that is in use by this ConnectionFactory.

**Returns**

a pointer to this objects **RedeliveryPolicy** (p. 3267).

**6.23.3.24 unsigned int activemq::core::ActiveMQConnection::getSendTimeout ( ) const**

Gets the assigned send timeout for this Connector.

**Returns**

the send timeout configured in the connection uri

**6.23.3.25 transport::Transport& activemq::core::ActiveMQConnection::getTransport ( ) const**

Gets a reference to this object's Transport instance.

**Returns**

a reference to the Transport that is in use by this Connection.

**6.23.3.26 const std::string& activemq::core::ActiveMQConnection::getUsername ( ) const**

Gets the username that this factory will use when creating a new connection instance.

**Returns**

username string, "" for default credentials

**6.23.3.27 bool activemq::core::ActiveMQConnection::isAlwaysSyncSend ( ) const**

Gets if the Connection should always send things Synchronously.

**Returns**

true if sends should always be Synchronous.

**6.23.3.28** `bool activemq::core::ActiveMQConnection::isClosed ( ) const` `[inline]`

Checks if this connection has been closed.

**Returns**

true if the connection is closed

**6.23.3.29** `bool activemq::core::ActiveMQConnection::isDispatchAsync ( ) const`

**Returns**

The value of the dispatch asynchronously option sent to the broker.

**6.23.3.30** `bool activemq::core::ActiveMQConnection::isStarted ( ) const` `[inline]`

Check if this connection has been started.

**Returns**

true if the start method has been called.

**6.23.3.31** `bool activemq::core::ActiveMQConnection::isTransportFailed ( ) const`  
`[inline]`

Checks if the Connection's Transport has failed.

**Returns**

true if the Connection's Transport has failed.

**6.23.3.32** `bool activemq::core::ActiveMQConnection::isUseAsyncSend ( ) const`

Gets if the useAsyncSend option is set.

**Returns**

true if on false if not.

**6.23.3.33** `bool activemq::core::ActiveMQConnection::isUseCompression ( ) const`

Gets if the Connection is configured for Message body compression.

**Returns**

if the Message body will be Compressed or not.

**6.23.3.34** `virtual void activemq::core::ActiveMQConnection::onCommand ( const Pointer< commands::Command > & command ) [virtual]`

Event handler for the receipt of a non-response command from the transport.

#### Parameters

<i>command</i>	the received command object.
----------------	------------------------------

Implements `activemq::transport::TransportListener` (p. 4014).

**6.23.3.35** `void activemq::core::ActiveMQConnection::oneway ( Pointer< commands::Command > command ) throw ( activemq::exceptions::ActiveMQException )`

Sends a oneway message.

#### Parameters

<i>command</i>	The message to send.
----------------	----------------------

#### Exceptions

<i>ConnectorException</i>	if not currently connected, or if the operation fails for any reason.
---------------------------	---

**6.23.3.36** `virtual void activemq::core::ActiveMQConnection::onException ( const decaf::lang::Exception & ex ) [virtual]`

Event handler for an exception from a command transport.

#### Parameters

<i>ex</i>	The exception.
-----------	----------------

Implements `activemq::transport::TransportListener` (p. 4015).

**6.23.3.37** `virtual void activemq::core::ActiveMQConnection::removeDispatcher ( const Pointer< commands::ConsumerId > & consumer ) throw ( cms::CMSException ) [virtual]`

Removes the dispatcher for a consumer.

#### Parameters

<i>consumer</i>	- The consumer for which to remove the dispatcher.
-----------------	--

**6.23.3.38** `virtual void activemq::core::ActiveMQConnection::removeProducer ( const Pointer< commands::ProducerId > & producerId ) throw ( cms::CMSException )`  
[virtual]

Removes an active Producer to the Set of known producers.

#### Parameters

<i>producerId</i>	- The ProducerId to remove from the the known set.
-------------------	--

**6.23.3.39** `virtual void activemq::core::ActiveMQConnection::removeSession ( ActiveMQSession * session ) throw ( cms::CMSException )` [virtual]

Removes the session resources for the given session instance.

#### Parameters

<i>session</i>	The session to be unregistered from this connection.
----------------	--

**6.23.3.40** `void activemq::core::ActiveMQConnection::removeTransportListener ( transport::TransportListener * transportListener )`

Removes a registered TransportListener from the Connection's set of Transport listeners, this listener will no longer receive any Transport related events.

The caller is responsible for freeing the listener in all cases.

#### Parameters

<i>transportListener</i>	The pointer to the TransportListener to remove from the set of listeners.
--------------------------	---

**6.23.3.41** `virtual void activemq::core::ActiveMQConnection::sendPullRequest ( const commands::ConsumerInfo * consumer, long long timeout ) throw ( exceptions::ActiveMQException )` [virtual]

If supported sends a message pull request to the service provider asking for the delivery of a new message.

This is used in the case where the service provider has been configured with a zero prefetch or is only capable of delivering messages on a pull basis.

#### Parameters

<i>consumer</i>	- the ConsumerInfo for the requesting Consumer.
<i>timeout</i>	- the time that the client is willing to wait.

**6.23.3.42 void activemq::core::ActiveMQConnection::setAlwaysSyncSend ( bool *value* )**

Sets if the Connection should always send things Synchronously.

**Parameters**

<i>value</i>	true if sends should always be Synchronous.
--------------	---

**6.23.3.43 void activemq::core::ActiveMQConnection::setBrokerURL ( const std::string & *brokerURL* )**

Sets the Broker URL that should be used when creating a new connection instance.

**Parameters**

<i>brokerURL</i>	string
------------------	--------

**6.23.3.44 virtual void activemq::core::ActiveMQConnection::setClientID ( const std::string & *clientID* ) [virtual]**

Sets the client identifier for this connection.

The preferred way to assign a CMS client's client identifier is for it to be configured in a client-specific **ConnectionFactory** (p. 1361) object and transparently assigned to the **Connection** (p. 1296) object it creates.

If a client sets the client identifier explicitly, it must do so immediately after it creates the connection and before any other action on the connection is taken. After this point, setting the client identifier is a programming error that should throw an **IllegalStateException** (p. 2059).

**Parameters**

<i>clientID</i>	The unique client identifier to assign to the <b>Connection</b> (p. 1296).
-----------------	--

**Exceptions**

<b>CMSException</b> (p. 1190)	if the provider fails to set the client id due to some internal error.
<b>InvalidClientIDException</b>	if the id given is somehow invalid or is a duplicate.
<b>IllegalStateException</b> (p. 2059)	if the client tries to set the id after a <b>Connection</b> (p. 1296) method has been called.

Implements **cms::Connection** (p. 1300).

**6.23.3.45 void activemq::core::ActiveMQConnection::setCloseTimeout ( unsigned int *timeout* )**

Sets the close timeout to use when sending the disconnect request.

**Parameters**

<i>timeout</i>	- The time to wait for a close message.
----------------	---

**6.23.3.46 void activemq::core::ActiveMQConnection::setDefaultClientId ( const std::string & *clientId* )**

Sets the Client Id.

**Parameters**

<i>clientId</i>	- The new clientId value.
-----------------	---------------------------

**6.23.3.47 void activemq::core::ActiveMQConnection::setDispatchAsync ( bool *value* )**

Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.

For slow consumers set it to true so that dispatching will not block fast consumers. .

**Parameters**

<i>value</i>	The value of the dispatch asynchronously option sent to the broker.
--------------	---

**6.23.3.48 virtual void activemq::core::ActiveMQConnection::setExceptionListener ( cms::ExceptionListener \* *listener* ) [virtual]**

Sets the registered Exception Listener for this connection.

**Parameters**

<i>listener</i>	pointer to and ExceptionListener
-----------------	----------------------------------

Implements **cms::Connection** (p. 1300).

**6.23.3.49 void activemq::core::ActiveMQConnection::setPassword ( const std::string & *password* )**

Sets the password that should be used when creating a new connection.

**Parameters**

<i>password</i>	string
-----------------	--------

**6.23.3.50 void activemq::core::ActiveMQConnection::setPrefetchPolicy ( PrefetchPolicy \* policy )**

Sets the **PrefetchPolicy** (p. 3062) instance that this factory should use when it creates new Connection instances.

The **PrefetchPolicy** (p. 3062) passed becomes the property of the factory and will be deleted when the factory is destroyed.

**Parameters**

<i>policy</i>	The new <b>PrefetchPolicy</b> (p. 3062) that the ConnectionFactory should clone for Connections.
---------------	--

**6.23.3.51 void activemq::core::ActiveMQConnection::setProducerWindowSize ( unsigned int windowSize )**

Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.

**Parameters**

<i>windowSize</i>	- The size in bytes of the Producers memory window.
-------------------	---

**6.23.3.52 void activemq::core::ActiveMQConnection::setRedeliveryPolicy ( RedeliveryPolicy \* policy )**

Sets the **RedeliveryPolicy** (p. 3267) instance that this factory should use when it creates new Connection instances.

The **RedeliveryPolicy** (p. 3267) passed becomes the property of the factory and will be deleted when the factory is destroyed.

**Parameters**

<i>policy</i>	The new <b>RedeliveryPolicy</b> (p. 3267) that the ConnectionFactory should clone for Connections.
---------------	--

**6.23.3.53 void activemq::core::ActiveMQConnection::setSendTimeout ( unsigned int timeout )**

Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.

**Parameters**

<i>timeout</i>	- The time to wait for a response.
----------------	------------------------------------

**6.23.3.54** `void activemq::core::ActiveMQConnection::setTransportInterruptProcessingComplete ( )`

Indicates that a Connection resource that is processing the transportInterrupted event has completed.

**6.23.3.55** `void activemq::core::ActiveMQConnection::setUseAsyncSend ( bool value )`

Sets the useAsyncSend option.

#### Parameters

<i>value</i>	- true to activate, false to disable.
--------------	---------------------------------------

**6.23.3.56** `void activemq::core::ActiveMQConnection::setUseCompression ( bool value )`

Sets whether Message body compression is enabled.

#### Parameters

<i>value</i>	Boolean indicating if Message body compression is enabled.
--------------	--

**6.23.3.57** `void activemq::core::ActiveMQConnection::setUsername ( const std::string & username )`

Sets the username that should be used when creating a new connection.

#### Parameters

<i>username</i>	string
-----------------	--------

**6.23.3.58** `virtual void activemq::core::ActiveMQConnection::start ( ) throw ( cms::CMSException ) [virtual]`

Starts or (restarts) a connections delivery of incoming messages.

#### Exceptions

<i>CMSException</i>
---------------------

Implements **cms::Startable** (p. 3692).



**6.23.3.59** `virtual void activemq::core::ActiveMQConnection::stop ( ) throw ( cms::CMSException )` [virtual]

Stop the flow of incoming messages.

#### Exceptions

<i>CMSException</i>	
---------------------	--

Implements **cms::Stoppable** (p. 3756).

**6.23.3.60** `void activemq::core::ActiveMQConnection::syncRequest ( Pointer< commands::Command > command, unsigned int timeout = 0 ) throw ( activemq::exceptions::ActiveMQException )`

Sends a synchronous request and returns the response from the broker.

Converts any error responses into an exception.

#### Parameters

<i>command</i>	The request command.
<i>timeout</i>	The time to wait for a response, default is zero or infinite.

#### Exceptions

<i>ConnectorException</i>	thrown if an error response was received from the broker, or if any other error occurred.
---------------------------	---

**6.23.3.61** `virtual void activemq::core::ActiveMQConnection::transportInterrupted ( )` [virtual]

The transport has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 4015).

**6.23.3.62** `virtual void activemq::core::ActiveMQConnection::transportResumed ( )` [virtual]

The transport has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 4015).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConnection.h`

## 6.24 activemq::core::ActiveMQConnectionFactory Class Reference

```
#include <src/main/activemq/core/ActiveMQConnectionFactory.h>
```

Inheritance diagram for activemq::core::ActiveMQConnectionFactory:

### Public Member Functions

- **ActiveMQConnectionFactory** ()
- **ActiveMQConnectionFactory** (const std::string &url, const std::string &username="", const std::string &password="")  
*Constructor.*
- virtual ~**ActiveMQConnectionFactory** ()
- virtual **cms::Connection** \* **createConnection** () throw ( cms::CMSException )  
*Creates a connection with the default user identity.*
- virtual **cms::Connection** \* **createConnection** (const std::string &username, const std::string &password) throw ( cms::CMSException )  
*Creates a connection with the specified user identity.*
- virtual **cms::Connection** \* **createConnection** (const std::string &username, const std::string &password, const std::string &clientId) throw ( cms::CMSException )  
*Creates a connection with the specified user identity.*
- void **setUsername** (const std::string &username)  
*Sets the username that should be used when creating a new connection.*
- const std::string & **getUsername** () const  
*Gets the username that this factory will use when creating a new connection instance.*
- void **setPassword** (const std::string &password)  
*Sets the password that should be used when creating a new connection.*
- const std::string & **getPassword** () const  
*Gets the password that this factory will use when creating a new connection instance.*
- std::string **getClientId** () const  
*Gets the Configured Client Id.*
- void **setClientId** (const std::string &clientId)  
*Sets the Client Id.*

- void **setBrokerURL** (const std::string &brokerURL)  
*Sets the Broker URL that should be used when creating a new connection instance.*
- const std::string & **getBrokerURL** () const  
*Gets the Broker URL that this factory will use when creating a new connection instance.*
- void **setExceptionListener** (cms::ExceptionListener \*listener)  
*Set an CMS ExceptionListener that will be set on eat connection once it has been created.*
- cms::ExceptionListener \* **getExceptionListener** () const  
*Returns the currently set ExceptionListener that will be set on any new Connection instance that is created by this factory.*
- void **setPrefetchPolicy** (PrefetchPolicy \*policy)  
*Sets the **PrefetchPolicy** (p. 3062) instance that this factory should use when it creates new Connection instances.*
- PrefetchPolicy \* **getPrefetchPolicy** () const  
*Gets the pointer to the current **PrefetchPolicy** (p. 3062) that is in use by this ConnectionFactory.*
- void **setRedeliveryPolicy** (RedeliveryPolicy \*policy)  
*Sets the **RedeliveryPolicy** (p. 3267) instance that this factory should use when it creates new Connection instances.*
- RedeliveryPolicy \* **getRedeliveryPolicy** () const  
*Gets the pointer to the current **RedeliveryPolicy** (p. 3267) that is in use by this ConnectionFactory.*
- bool **isDispatchAsync** () const
- void **setDispatchAsync** (bool value)  
*Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.*
- bool **isAlwaysSyncSend** () const  
*Gets if the Connection should always send things Synchronously.*
- void **setAlwaysSyncSend** (bool value)  
*Sets if the Connection should always send things Synchronously.*
- bool **isUseAsyncSend** () const  
*Gets if the useAsyncSend option is set.*
- void **setUseAsyncSend** (bool value)  
*Sets the useAsyncSend option.*

- **bool isUseCompression () const**  
*Gets if the Connection is configured for Message body compression.*
- **void setUseCompression (bool value)**  
*Sets whether Message body compression is enabled.*
- **unsigned int getSendTimeout () const**  
*Gets the assigned send timeout for this Connector.*
- **void setSendTimeout (unsigned int timeout)**  
*Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.*
- **unsigned int getCloseTimeout () const**  
*Gets the assigned close timeout for this Connector.*
- **void setCloseTimeout (unsigned int timeout)**  
*Sets the close timeout to use when sending the disconnect request.*
- **unsigned int getProducerWindowSize () const**  
*Gets the configured producer window size for Producers that are created from this connector.*
- **void setProducerWindowSize (unsigned int windowSize)**  
*Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.*

### Static Public Member Functions

- **static cms::Connection \* createConnection** (const std::string &url, const std::string &username, const std::string &password, const std::string &clientId="") throw ( cms::CMSException )  
*Creates a connection with the specified user identity.*

### Static Public Attributes

- **static const std::string DEFAULT\_URI**

### 6.24.1 Constructor & Destructor Documentation

6.24.1.1 `activemq::core::ActiveMQConnectionFactory::ActiveMQConnectionFactory ( )`

6.24.1.2 `activemq::core::ActiveMQConnectionFactory::ActiveMQConnectionFactory ( const std::string & url, const std::string & username = " ", const std::string & password = " " )`

Constructor.

#### Parameters

<i>url</i>	the URL of the Broker we are connecting to.
<i>username</i>	to authenticate with, defaults to ""
<i>password</i>	to authenticate with, defaults to ""

6.24.1.3 `virtual activemq::core::ActiveMQConnectionFactory::~~ActiveMQConnectionFactory ( ) [virtual]`

### 6.24.2 Member Function Documentation

6.24.2.1 `virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection ( ) throw ( cms::CMSEException ) [virtual]`

Creates a connection with the default user identity.

The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called.

#### Returns

a Connection Pointer

#### Exceptions

<i>CMSEException</i>	
----------------------	--

Implements `cms::ConnectionFactory` (p. 1364).

6.24.2.2 `virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection ( const std::string & username, const std::string & password ) throw ( cms::CMSEException ) [virtual]`

Creates a connection with the specified user identity.

The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless createConnection will continue to use the default values that were set in the Constructor.

**Parameters**

<i>username</i>	to authenticate with
<i>password</i>	to authenticate with

**Returns**

a Connection Pointer

**Exceptions**

<i>CMSEException</i>	
----------------------	--

Implements **cms::ConnectionFactory** (p. 1363).

**6.24.2.3** `static cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection ( const std::string & url, const std::string & username, const std::string & password, const std::string & clientId = " " ) throw ( cms::CMSEException ) [static]`

Creates a connection with the specified user identity.

The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called.

**Parameters**

<i>url</i>	the URL of the Broker we are connecting to.
<i>username</i>	to authenticate with
<i>password</i>	to authenticate with
<i>clientId</i>	to assign to connection, defaults to ""

**Exceptions**

<i>CMSEException.</i>	
-----------------------	--

**6.24.2.4** `virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection ( const std::string & username, const std::string & password, const std::string & clientId ) throw ( cms::CMSEException ) [virtual]`

Creates a connection with the specified user identity.

The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless createConnection will continue to use the default values that were set in the Constructor.

**Parameters**

<i>username</i>	to authenticate with
<i>password</i>	to authenticate with
<i>clientId</i>	to assign to connection if "" then a random cleint Id is created for this connection.

**Returns**

a Connection Pointer

**Exceptions**

<i>CMSEException</i>	
----------------------	--

Implements **cms::ConnectionFactory** (p. 1364).

**6.24.2.5** `const std::string& activemq::core::ActiveMQConnectionFactory::getBrokerURL ( ) const`

Gets the Broker URL that this factory will use when creating a new connection instance.

**Returns**

brokerURL string

**6.24.2.6** `std::string activemq::core::ActiveMQConnectionFactory::getClientId ( ) const`

Gets the Configured Client Id.

**Returns**

the clientId.

**6.24.2.7** `unsigned int activemq::core::ActiveMQConnectionFactory::getCloseTimeout ( ) const`

Gets the assigned close timeout for this Connector.

**Returns**

the close timeout configured in the connection uri

**6.24.2.8** `cms::ExceptionListener* activemq::core::ActiveMQConnectionFactory::getExceptionListener ( ) const`

Returns the currently set ExceptionListener that will be set on any new Connection instance that is created by this factory.

**Returns**

a pointer to a CMS ExceptionListener instance or NULL if not set.

**6.24.2.9** `const std::string& activemq::core::ActiveMQConnectionFactory::getPassword ( ) const`

Gets the password that this factory will use when creating a new connection instance.

**Returns**

password string, "" for default credentials

**6.24.2.10** `PrefetchPolicy* activemq::core::ActiveMQConnectionFactory::getPrefetchPolicy ( ) const`

Gets the pointer to the current **PrefetchPolicy** (p. 3062) that is in use by this ConnectionFactory.

**Returns**

a pointer to this objects **PrefetchPolicy** (p. 3062).

**6.24.2.11** `unsigned int activemq::core::ActiveMQConnectionFactory::getProducerWindowSize ( ) const`

Gets the configured producer window size for Producers that are created from this connector.

This only applies if there is no send timeout and the producer is able to send asynchronously.

**Returns**

size in bytes of messages that this producer can produce before it must block and wait for ProducerAck messages to free resources.

**6.24.2.12** `RedeliveryPolicy* activemq::core::ActiveMQConnectionFactory::getRedeliveryPolicy ( ) const`

Gets the pointer to the current **RedeliveryPolicy** (p. 3267) that is in use by this ConnectionFactory.

**Returns**

a pointer to this objects **RedeliveryPolicy** (p. 3267).

**6.24.2.13** `unsigned int activemq::core::ActiveMQConnectionFactory::getSendTimeout ( ) const`

Gets the assigned send timeout for this Connector.



**Returns**

the send timeout configured in the connection uri

**6.24.2.14** `const std::string& activemq::core::ActiveMQConnectionFactory::getUsername ( ) const`

Gets the username that this factory will use when creating a new connection instance.

**Returns**

username string, "" for default credentials

**6.24.2.15** `bool activemq::core::ActiveMQConnectionFactory::isAlwaysSyncSend ( ) const`

Gets if the Connection should always send things Synchronously.

**Returns**

true if sends should always be Synchronous.

**6.24.2.16** `bool activemq::core::ActiveMQConnectionFactory::isDispatchAsync ( ) const`

**Returns**

The value of the dispatch asynchronously option sent to the broker.

**6.24.2.17** `bool activemq::core::ActiveMQConnectionFactory::isUseAsyncSend ( ) const`

Gets if the useAsyncSend option is set.

**Returns**

true if on false if not.

**6.24.2.18** `bool activemq::core::ActiveMQConnectionFactory::isUseCompression ( ) const`

Gets if the Connection is configured for Message body compression.

**Returns**

if the Message body will be Compressed or not.

**6.24.2.19 void activemq::core::ActiveMQConnectionFactory::setAlwaysSyncSend ( bool *value* )**

Sets if the Connection should always send things Synchronously.

**Parameters**

<i>value</i>	true if sends should always be Synchronous.
--------------	---

**6.24.2.20 void activemq::core::ActiveMQConnectionFactory::setBrokerURL ( const std::string & *brokerURL* )**

Sets the Broker URL that should be used when creating a new connection instance.

**Parameters**

<i>brokerURL</i>	string
------------------	--------

**6.24.2.21 void activemq::core::ActiveMQConnectionFactory::setClientId ( const std::string & *clientId* )**

Sets the Client Id.

**Parameters**

<i>clientId</i>	- The new clientId value.
-----------------	---------------------------

**6.24.2.22 void activemq::core::ActiveMQConnectionFactory::setCloseTimeout ( unsigned int *timeout* )**

Sets the close timeout to use when sending the disconnect request.

**Parameters**

<i>timeout</i>	- The time to wait for a close message.
----------------	---

**6.24.2.23 void activemq::core::ActiveMQConnectionFactory::setDispatchAsync ( bool *value* )**

Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.

For slow consumers set it to true so that dispatching will not block fast consumers. .

**Parameters**

<i>value</i>	The value of the dispatch asynchronously option sent to the broker.
--------------	---

#### 6.24.2.24 void activemq::core::ActiveMQConnectionFactory::setExceptionListener ( cms::ExceptionListener \* *listener* )

Set an CMS ExceptionListener that will be set on eat connection once it has been created.

The factory does not take ownership of this pointer, the client must ensure that its lifetime is scoped to the connection that it is applied to.

##### Parameters

<i>listener</i>	The listener to set on the connection or NULL for no listener.
-----------------	--

#### 6.24.2.25 void activemq::core::ActiveMQConnectionFactory::setPassword ( const std::string & *password* )

Sets the password that should be used when creating a new connection.

##### Parameters

<i>password</i>	string
-----------------	--------

#### 6.24.2.26 void activemq::core::ActiveMQConnectionFactory::setPrefetchPolicy ( PrefetchPolicy \* *policy* )

Sets the **PrefetchPolicy** (p. 3062) instance that this factory should use when it creates new Connection instances.

The **PrefetchPolicy** (p. 3062) passed becomes the property of the factory and will be deleted when the factory is destroyed.

##### Parameters

<i>policy</i>	The new <b>PrefetchPolicy</b> (p. 3062) that the ConnectionFactory should clone for Connections.
---------------	--

#### 6.24.2.27 void activemq::core::ActiveMQConnectionFactory::setProducerWindowSize ( unsigned int *windowSize* )

Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.

##### Parameters

<i>windowSize</i>	- The size in bytes of the Producers memory window.
-------------------	---

#### 6.24.2.28 void activemq::core::ActiveMQConnectionFactory::setRedeliveryPolicy ( **RedeliveryPolicy** \* *policy* )

Sets the **RedeliveryPolicy** (p. 3267) instance that this factory should use when it creates new Connection instances.

The **RedeliveryPolicy** (p. 3267) passed becomes the property of the factory and will be deleted when the factory is destroyed.

##### Parameters

<i>policy</i>	The new <b>RedeliveryPolicy</b> (p. 3267) that the ConnectionFactory should clone for Connections.
---------------	--

#### 6.24.2.29 void activemq::core::ActiveMQConnectionFactory::setSendTimeout ( unsigned int *timeout* )

Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.

##### Parameters

<i>timeout</i>	- The time to wait for a response.
----------------	------------------------------------

#### 6.24.2.30 void activemq::core::ActiveMQConnectionFactory::setUseAsyncSend ( bool *value* )

Sets the useAsyncSend option.

##### Parameters

<i>value</i>	- true to activate, false to disable.
--------------	---------------------------------------

#### 6.24.2.31 void activemq::core::ActiveMQConnectionFactory::setUseCompression ( bool *value* )

Sets whether Message body compression is enabled.

##### Parameters

<i>value</i>	Boolean indicating if Message body compression is enabled.
--------------	--

#### 6.24.2.32 void activemq::core::ActiveMQConnectionFactory::setUsername ( const std::string & *username* )

Sets the username that should be used when creating a new connection.

**Parameters**

<i>username</i>	string
-----------------	--------

**6.24.3 Field Documentation****6.24.3.1 const std::string activemq::core::ActiveMQConnectionFactory::DEFAULT\_URI [static]**

The documentation for this class was generated from the following file:

- src/main/activemq/core/ActiveMQConnectionFactory.h

**6.25 activemq::core::ActiveMQConnectionMetaData Class Reference**

This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 260) class.

```
#include <src/main/activemq/core/ActiveMQConnectionMetaData.h>
```

Inheritance diagram for activemq::core::ActiveMQConnectionMetaData:

**Public Member Functions**

- **ActiveMQConnectionMetaData ()**
- virtual **~ActiveMQConnectionMetaData ()**
- virtual std::string **getCMSVersion ()** const throw ( cms::CMSException )  
*Gets the CMS API version.*
- virtual int **getCMSMajorVersion ()** const throw ( cms::CMSException )  
*Gets the CMS major version number.*
- virtual int **getCMSMinorVersion ()** const throw ( cms::CMSException )  
*Gets the CMS minor version number.*
- virtual std::string **getCMSProviderName ()** const throw ( cms::CMSException )  
*Gets the CMS provider name.*
- virtual std::string **getProviderVersion ()** const throw ( cms::CMSException )  
*Gets the CMS provider version.*
- virtual int **getProviderMajorVersion ()** const throw ( cms::CMSException )  
*Gets the CMS provider major version number.*

- virtual int **getProviderMinorVersion** () const throw ( cms::CMSEException )  
*Gets the CMS provider minor version number.*
- virtual std::vector< std::string > **getCMSXPropertyNames** () const throw ( cms::CMSEException )  
*Gets an Vector of the CMSX property names.*

### 6.25.1 Detailed Description

This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 260) class.

#### Since

3.0

### 6.25.2 Constructor & Destructor Documentation

6.25.2.1 **activemq::core::ActiveMQConnectionMetaData::ActiveMQConnectionMetaData** ( )

6.25.2.2 **virtual activemq::core::ActiveMQConnectionMetaData::~~ActiveMQConnectionMetaData** ( ) [virtual]

### 6.25.3 Member Function Documentation

6.25.3.1 **virtual int activemq::core::ActiveMQConnectionMetaData::getCMSMajorVersion** ( ) const throw ( cms::CMSEException ) [virtual]

Gets the CMS major version number.

#### Returns

the CMS API major version number

#### Exceptions

<i>CMSEException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
----------------------	--

Implements **cms::ConnectionMetaData** (p. 1426).

6.25.3.2 **virtual int activemq::core::ActiveMQConnectionMetaData::getCMSMinorVersion** ( ) const throw ( cms::CMSEException ) [virtual]

Gets the CMS minor version number.

**Returns**

the CMS API minor version number

**Exceptions**

<i>CMSEException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
----------------------	--

Implements **cms::ConnectionMetaData** (p. 1427).

**6.25.3.3** `virtual std::string activemq::core::ActiveMQConnectionMetaData::getCMSProviderName ( ) const throw ( cms::CMSEException ) [virtual]`

Gets the CMS provider name.

**Returns**

the CMS provider name

**Exceptions**

<i>CMSEException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
----------------------	--

Implements **cms::ConnectionMetaData** (p. 1427).

**6.25.3.4** `virtual std::string activemq::core::ActiveMQConnectionMetaData::getCMSVersion ( ) const throw ( cms::CMSEException ) [virtual]`

Gets the CMS API version.

**Returns**

the CMS API Version in String form.

**Exceptions**

<i>CMSEException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
----------------------	--

Implements **cms::ConnectionMetaData** (p. 1427).

**6.25.3.5** `virtual std::vector<std::string> activemq::core::ActiveMQConnectionMetaData::getCMSXPropertyNames ( ) const throw ( cms::CMSEException ) [virtual]`

Gets an Vector of the CMSX property names.

**Returns**

an Vector of CMSX property names

**Exceptions**

<i>CMSEException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
----------------------	--

Implements **cms::ConnectionMetaData** (p. 1428).

**6.25.3.6** `virtual int activemq::core::ActiveMQConnectionMetaData::getProviderMajorVersion ( )  
const throw ( cms::CMSEException ) [virtual]`

Gets the CMS provider major version number.

**Returns**

the CMS provider major version number

**Exceptions**

<i>CMSEException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
----------------------	--

Implements **cms::ConnectionMetaData** (p. 1428).

**6.25.3.7** `virtual int activemq::core::ActiveMQConnectionMetaData::getProviderMinorVersion ( )  
const throw ( cms::CMSEException ) [virtual]`

Gets the CMS provider minor version number.

**Returns**

the CMS provider minor version number

**Exceptions**

<i>CMSEException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
----------------------	--

Implements **cms::ConnectionMetaData** (p. 1428).

**6.25.3.8** `virtual std::string activemq::core::ActiveMQConnectionMetaData::getProviderVersion ( )  
const throw ( cms::CMSEException ) [virtual]`

Gets the CMS provider version.



**Returns**

the CMS provider version

**Exceptions**

<i>CMSException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
---------------------	--

Implements **cms::ConnectionMetaData** (p. 1429).

The documentation for this class was generated from the following file:

- src/main/activemq/core/ActiveMQConnectionMetaData.h

## 6.26 activemq::core::ActiveMQConstants Class Reference

Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values.

```
#include <src/main/activemq/core/ActiveMQConstants.h>
```

**Data Structures**

- class **StaticInitializer**

**Public Types**

- enum **TransactionState** {  
TRANSACTION\_STATE\_BEGIN = 0, TRANSACTION\_STATE\_PREPARE = 1, TRANSACTION\_STATE\_COMMITONEPHASE = 2, TRANSACTION\_STATE\_COMMITTWO PHASE = 3,  
TRANSACTION\_STATE\_ROLLBACK = 4, TRANSACTION\_STATE\_RECOVER = 5, TRANSACTION\_STATE\_FORGET = 6, TRANSACTION\_STATE\_END = 7 }
- enum **DestinationActions** { DESTINATION\_ADD\_OPERATION = 0, DESTINATION\_REMOVE\_OPERATION = 1 }
- enum **AckType** {  
ACK\_TYPE\_DELIVERED = 0, ACK\_TYPE\_POISON = 1, ACK\_TYPE\_CONSUMED = 2, ACK\_TYPE\_REDELIVERED = 3,  
ACK\_TYPE\_INDIVIDUAL = 4 }
- enum **DestinationOption** {  
CONSUMER\_PREFETCHSIZE, CONSUMER\_MAXPENDINGMSGLIMIT, CONSUMER\_NOLOCAL, CONSUMER\_DISPATCHASYNC,  
CONSUMER\_RETROACTIVE, CONSUMER\_SELECTOR, CONSUMER\_EXCLUSIVE, CONSUMER\_PRIORITY,

**NUM\_OPTIONS }**

*These values represent the options that can be appended to an Destination name, i.e.*

- enum **URIParam** {  
**CONNECTION\_SENDTIMEOUT, CONNECTION\_PRODUCERWINDOWSIZE,**  
**CONNECTION\_CLOSETIMEOUT, CONNECTION\_ALWAYSSEND,**  
**CONNECTION\_USEASYNCSEND, CONNECTION\_USECOMPRESSION,**  
**CONNECTION\_DISPATCHASYNC, PARAM\_USERNAME,**  
**PARAM\_PASSWORD, PARAM\_CLIENTID, NUM\_PARAMS }**

*These values represent the parameters that can be added to the connection URI that affect the ActiveMQ Core API.*

**Static Public Member Functions**

- static const std::string & **toString** (const **DestinationOption** option)
- static **DestinationOption** **toDestinationOption** (const std::string &option)
- static const std::string & **toString** (const **URIParam** option)
- static **URIParam** **toURIOption** (const std::string &option)

**6.26.1 Detailed Description**

Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back an forth between string and enum values.

**6.26.2 Member Enumeration Documentation****6.26.2.1 enum activemq::core::ActiveMQConstants::AckType**

**Enumerator:**

***ACK\_TYPE\_DELIVERED***  
***ACK\_TYPE\_POISON***  
***ACK\_TYPE\_CONSUMED***  
***ACK\_TYPE\_REDELIVERED***  
***ACK\_TYPE\_INDIVIDUAL***

**6.26.2.2 enum activemq::core::ActiveMQConstants::DestinationActions**

**Enumerator:**

***DESTINATION\_ADD\_OPERATION***  
***DESTINATION\_REMOVE\_OPERATION***

**6.26.2.3 enum activemq::core::ActiveMQConstants::DestinationOption**

These values represent the options that can be appended to an Destination name, i.e.

/topic/foo?consumer.exclusive=true

**Enumerator:**

*CONSUMER\_PREFETCHSIZE*  
*CONSUMER\_MAXPENDINGMSGLIMIT*  
*CONSUMER\_NOLOCAL*  
*CONSUMER\_DISPATCHASYNC*  
*CONSUMER\_RETROACTIVE*  
*CONSUMER\_SELECTOR*  
*CONSUMER\_EXCLUSIVE*  
*CONSUMER\_PRIORITY*  
*NUM\_OPTIONS*

**6.26.2.4 enum activemq::core::ActiveMQConstants::TransactionState****Enumerator:**

*TRANSACTION\_STATE\_BEGIN*  
*TRANSACTION\_STATE\_PREPARE*  
*TRANSACTION\_STATE\_COMMITONEPHASE*  
*TRANSACTION\_STATE\_COMMITTWOPHASE*  
*TRANSACTION\_STATE\_ROLLBACK*  
*TRANSACTION\_STATE\_RECOVER*  
*TRANSACTION\_STATE\_FORGET*  
*TRANSACTION\_STATE\_END*

**6.26.2.5 enum activemq::core::ActiveMQConstants::URIParam**

These values represent the parameters that can be added to the connection URI that affect the ActiveMQ Core API.

**Enumerator:**

*CONNECTION\_SENDTIMEOUT*  
*CONNECTION\_PRODUCERWINDOWSIZE*  
*CONNECTION\_CLOSETIMEOUT*  
*CONNECTION\_ALWAYSYNCSEND*  
*CONNECTION\_USEASYNCSEND*

*CONNECTION\_USECOMPRESSION*

*CONNECTION\_DISPATCHASYNC*

*PARAM\_USERNAME*

*PARAM\_PASSWORD*

*PARAM\_CLIENTID*

*NUM\_PARAMS*

### 6.26.3 Member Function Documentation

6.26.3.1 static **DestinationOption** activemq::core::ActiveMQConstants::toDestinationOption ( const std::string & *option* ) [inline, static]

6.26.3.2 static const std::string& activemq::core::ActiveMQConstants::toString ( const **DestinationOption** *option* ) [inline, static]

6.26.3.3 static const std::string& activemq::core::ActiveMQConstants::toString ( const **URIParam** *option* ) [inline, static]

6.26.3.4 static **URIParam** activemq::core::ActiveMQConstants::toURIOption ( const std::string & *option* ) [inline, static]

The documentation for this class was generated from the following file:

- src/main/activemq/core/ActiveMQConstants.h

## 6.27 activemq::core::ActiveMQConsumer Class Reference

```
#include <src/main/activemq/core/ActiveMQConsumer.h>
```

Inheritance diagram for activemq::core::ActiveMQConsumer:

### Public Member Functions

- **ActiveMQConsumer** (**ActiveMQSession** \*session, const **Pointer**< **commands::ConsumerId** > &id, const **Pointer**< **commands::ActiveMQDestination** > &destination, const std::string &name, const std::string &selector, int prefetch, int maxPendingMessageCount, bool noLocal, bool browser, bool dispatchAsync, **cms::MessageListener** \*listener)

*Constructor.*

- virtual ~**ActiveMQConsumer** ()
- virtual void **start** ()

*Starts the Consumer if not already started and not closed.*

- virtual void **stop** ()  
*Stops a Consumer, the Consumer will not deliver any messages that are dispatched to it until it is started again.*
- virtual void **close** () throw ( cms::CMSEException )  
*Closes the Consumer.*
- virtual **cms::Message** \* **receive** () throw ( cms::CMSEException )  
*Synchronously Receive a Message.*
- virtual **cms::Message** \* **receive** (int millisecs) throw ( cms::CMSEException )  
*Synchronously Receive a Message, time out after defined interval.*
- virtual **cms::Message** \* **receiveNoWait** () throw ( cms::CMSEException )  
*Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.*
- virtual void **setMessageListener** (cms::MessageListener \*listener) throw ( cms::CMSEException )  
*Sets the MessageListener that this class will send notifys on.*
- virtual **cms::MessageListener** \* **getMessageListener** () const throw ( cms::CMSEException )  
*Gets the MessageListener that this class will send events to.*
- virtual std::string **getMessageSelector** () const throw ( cms::CMSEException )  
*Gets this message consumer's message selector expression.*
- virtual void **acknowledge** (const **Pointer**< **commands::MessageDispatch** > &dispatch) throw ( cms::CMSEException )  
*Method called to acknowledge the message passed, called from a message when the mode is client ack.*
- virtual void **dispatch** (const **Pointer**< **MessageDispatch** > &message)  
*Called asynchronously by the session to dispatch a message.*
- void **acknowledge** () throw ( cms::CMSEException )  
*Method called to acknowledge all messages that have been received so far.*
- void **commit** () throw ( exceptions::ActiveMQException )  
*Called to Commit the current set of messages in this Transaction.*
- void **rollback** () throw ( exceptions::ActiveMQException )  
*Called to Roll back the current set of messages in this Transaction.*
- void **doClose** () throw ( exceptions::ActiveMQException )

*Performs the actual close operation on this consumer.*

- **const Pointer< commands::ConsumerInfo > & getConsumerInfo () const**  
*Get the Consumer information for this consumer.*
- **const Pointer< commands::ConsumerId > & getConsumerId () const**  
*Get the Consumer Id for this consumer.*
- **bool isClosed () const**
- **bool isSynchronizationRegistered () const**  
*Has this Consumer Transaction **Synchronization** (p. 3826) been added to the transaction.*
- **void setSynchronizationRegistered (bool value)**  
*Sets the **Synchronization** (p. 3826) Registered state of this consumer.*
- **bool iterate ()**  
*Deliver any pending messages to the registered MessageListener if there is one, return true if not all dispatched, or false if no listener or all pending messages have been dispatched.*
- **void deliverAcks () throw ( exceptions::ActiveMQException )**  
*Forces this consumer to send all pending acks to the broker.*
- **void clearMessagesInProgress ()**  
*Called on a Failover to clear any pending messages.*
- **void inProgressClearRequired ()**  
*Signals that a Failure occurred and that anything in-progress in the consumer should be cleared.*
- **long long getLastDeliveredSequenceId () const**  
*Gets the currently set Last Delivered Sequence Id.*
- **void setLastDeliveredSequenceId (long long value)**  
*Sets the value of the Last Delivered Sequence Id.*
- **int getMessageAvailableCount () const**
- **void setRedeliveryPolicy (RedeliveryPolicy \*policy)**  
*Sets the **RedeliveryPolicy** (p. 3267) this Consumer should use when a rollback is performed on a transacted Consumer.*
- **RedeliveryPolicy \* getRedeliveryPolicy () const**  
*Gets a pointer to this Consumer's Redelivery Policy object, the Consumer retains ownership of this pointer so the caller should not delete it.*

## Protected Member Functions

- **Pointer< MessageDispatch > dequeue** (long long timeout) throw ( cms::CMSException )  
*Used by synchronous receive methods to wait for messages to come in.*
- **void beforeMessageIsConsumed** (const Pointer< commands::MessageDispatch > &dispatch)  
*Pre-consume processing.*
- **void afterMessageIsConsumed** (const Pointer< commands::MessageDispatch > &dispatch, bool messageExpired)  
*Post-consume processing.*

## 6.27.1 Constructor & Destructor Documentation

- 6.27.1.1 **activemq::core::ActiveMQConsumer::ActiveMQConsumer** ( ActiveMQSession \* session, const Pointer< commands::ConsumerId > & id, const Pointer< commands::ActiveMQDestination > & destination, const std::string & name, const std::string & selector, int prefetch, int maxPendingMessageCount, bool noLocal, bool browser, bool dispatchAsync, cms::MessageListener \* listener )

Constructor.

- 6.27.1.2 **virtual activemq::core::ActiveMQConsumer::~~ActiveMQConsumer** ( )  
 [virtual]

## 6.27.2 Member Function Documentation

- 6.27.2.1 **virtual void activemq::core::ActiveMQConsumer::acknowledge** ( const Pointer< commands::MessageDispatch > & dispatch ) throw ( cms::CMSException )  
 [virtual]

Method called to acknowledge the message passed, called from a message when the mode is client ack.

### Parameters

<i>message</i>	the Message to Acknowledge
----------------	----------------------------

### Exceptions

<i>CMSException</i>	
---------------------	--

**6.27.2.2** `void activemq::core::ActiveMQConsumer::acknowledge ( ) throw ( cms::CMSException )`

Method called to acknowledge all messages that have been received so far.

#### Exceptions

<i>CMSException</i>	
---------------------	--

**6.27.2.3** `void activemq::core::ActiveMQConsumer::afterMessagelsConsumed ( const Pointer< commands::MessageDispatch > & dispatch, bool messageExpired ) [protected]`

Post-consume processing.

#### Parameters

<i>dispatch</i>	- the consumed message
<i>messageExpired</i>	- flag indicating if the message has expired.

**6.27.2.4** `void activemq::core::ActiveMQConsumer::beforeMessagelsConsumed ( const Pointer< commands::MessageDispatch > & dispatch ) [protected]`

Pre-consume processing.

#### Parameters

<i>dispatch</i>	- the message being consumed.
-----------------	-------------------------------

**6.27.2.5** `void activemq::core::ActiveMQConsumer::clearMessagesInProgress ( )`

Called on a Failover to clear any pending messages.

**6.27.2.6** `virtual void activemq::core::ActiveMQConsumer::close ( ) throw ( cms::CMSException ) [virtual]`

Closes the Consumer.

This will return all allocated resources and purge any outstanding messages. This method will block if there is a call to receive in progress, or a dispatch to a MessageListener in place

#### Exceptions

<i>CMSException</i>	
---------------------	--



Implements **cms::Closeable** (p. 1180).

**6.27.2.7** `void activemq::core::ActiveMQConsumer::commit ( ) throw ( exceptions::ActiveMQException )`

Called to Commit the current set of messages in this Transaction.

#### Exceptions

<i>ActiveMQException</i>	
--------------------------	--

**6.27.2.8** `void activemq::core::ActiveMQConsumer::deliverAcks ( ) throw ( exceptions::ActiveMQException )`

Forces this consumer to send all pending acks to the broker.

**6.27.2.9** `Pointer<MessageDispatch> activemq::core::ActiveMQConsumer::dequeue ( long long timeout ) throw ( cms::CMSException )` [protected]

Used by synchronous receive methods to wait for messages to come in.

#### Parameters

<i>timeout</i>	- The maximum number of milliseconds to wait before returning. If -1, it will block until a messages is received or this consumer is closed. If 0, will not block at all. If > 0, will wait at a maximum the specified number of milliseconds before returning.
----------------	---

#### Returns

the message, if received within the allotted time. Otherwise NULL.

#### Exceptions

<i>InvalidStateException</i>	if this consumer is closed upon entering this method.
------------------------------	---

**6.27.2.10** `virtual void activemq::core::ActiveMQConsumer::dispatch ( const Pointer< MessageDispatch > & message )` [virtual]

Called asynchronously by the session to dispatch a message.

#### Parameters

<i>message</i>	dispatch object pointer
----------------	-------------------------

Implements **activemq::core::Dispatcher** (p. 1841).

**6.27.2.11** `void activemq::core::ActiveMQConsumer::doClose ( ) throw ( exceptions::ActiveMQException )`

Performs the actual close operation on this consumer.

#### Exceptions

<i>ActiveMQException</i>
--------------------------

**6.27.2.12** `const Pointer<commands::ConsumerId>& activemq::core::ActiveMQConsumer::getConsumerId ( ) const [inline]`

Get the Consumer Id for this consumer.

#### Returns

Reference to a Consumer Id Object

**6.27.2.13** `const Pointer<commands::ConsumerInfo>& activemq::core::ActiveMQConsumer::getConsumerInfo ( ) const [inline]`

Get the Consumer information for this consumer.

#### Returns

Reference to a Consumer Info Object

**6.27.2.14** `long long activemq::core::ActiveMQConsumer::getLastDeliveredSequenceId ( ) const [inline]`

Gets the currently set Last Delivered Sequence Id.

#### Returns

long long containing the sequence id of the last delivered Message.

**6.27.2.15** `int activemq::core::ActiveMQConsumer::getMessageAvailableCount ( ) const`

#### Returns

the number of Message's this consumer is waiting to Dispatch.

**6.27.2.16** `virtual cms::MessageListener* activemq::core::ActiveMQConsumer::getMessageListener ( ) const throw ( cms::CMSException ) [inline, virtual]`

Gets the MessageListener that this class will send events to.

#### Returns

the currently registered MessageListener interface pointer.

Implements **cms::MessageConsumer** (p. 2675).

**6.27.2.17** `virtual std::string activemq::core::ActiveMQConsumer::getMessageSelector ( ) const throw ( cms::CMSException ) [virtual]`

Gets this message consumer's message selector expression.

#### Returns

This Consumer's selector expression or "".

#### Exceptions

<i>cms::CMSException</i> (p. 1190)	
---------------------------------------	--

Implements **cms::MessageConsumer** (p. 2676).

**6.27.2.18** `RedeliveryPolicy* activemq::core::ActiveMQConsumer::getRedeliveryPolicy ( ) const [inline]`

Gets a pointer to this Consumer's Redelivery Policy object, the Consumer retains ownership of this pointer so the caller should not delete it.

#### Returns

a Pointer to a **RedeliveryPolicy** (p. 3267) that is in use by this Consumer.

**6.27.2.19** `void activemq::core::ActiveMQConsumer::inProgressClearRequired ( )`

Signals that a Failure occurred and that anything in-progress in the consumer should be cleared.

**6.27.2.20** `bool activemq::core::ActiveMQConsumer::isClosed ( ) const [inline]`

#### Returns

if this Consumer has been closed.

**6.27.2.21** `bool activemq::core::ActiveMQConsumer::isSynchronizationRegistered ( ) const`  
`[inline]`

Has this Consumer Transaction **Synchronization** (p. 3826) been added to the transaction.

#### Returns

true if the synchronization has been added.

**6.27.2.22** `bool activemq::core::ActiveMQConsumer::iterate ( )`

Deliver any pending messages to the registered MessageListener if there is one, return true if not all dispatched, or false if no listener or all pending messages have been dispatched.

**6.27.2.23** `virtual cms::Message* activemq::core::ActiveMQConsumer::receive ( ) throw ( cms::CMSEException )` `[virtual]`

Synchronously Receive a Message.

#### Returns

new message

#### Exceptions

<i>CMSEException</i>
----------------------

Implements **cms::MessageConsumer** (p. 2676).

**6.27.2.24** `virtual cms::Message* activemq::core::ActiveMQConsumer::receive ( int millisecs ) throw ( cms::CMSEException )` `[virtual]`

Synchronously Receive a Message, time out after defined interval.

Returns null if nothing read.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait before returning
------------------	---

#### Returns

new message or null on timeout

#### Exceptions

<i>CMSEException</i>
----------------------

Implements **cms::MessageConsumer** (p. 2676).

**6.27.2.25** `virtual cms::Message* activemq::core::ActiveMQConsumer::receiveNoWait ( )  
throw ( cms::CMSEException ) [virtual]`

Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.

#### Returns

new message

#### Exceptions

<i>CMSEException</i>	
----------------------	--

Implements **cms::MessageConsumer** (p. 2677).

**6.27.2.26** `void activemq::core::ActiveMQConsumer::rollback ( ) throw ( exceptions::ActiveMQException )`

Called to Roll back the current set of messages in this Transaction.

#### Exceptions

<i>ActiveMQException</i>	
--------------------------	--

**6.27.2.27** `void activemq::core::ActiveMQConsumer::setLastDeliveredSequenceId ( long long  
value ) [inline]`

Sets the value of the Last Delivered Sequence Id.

#### Parameters

<i>value</i>	The new value to assign to the Last Delivered Sequence Id property.
--------------	---

**6.27.2.28** `virtual void activemq::core::ActiveMQConsumer::setMessageListener ( cms::MessageListener * listener ) throw ( cms::CMSEException )  
[virtual]`

Sets the MessageListener that this class will send notifs on.

#### Parameters

<i>listener</i>	MessageListener interface pointer
-----------------	-----------------------------------

Implements **cms::MessageConsumer** (p. 2677).

**6.27.2.29** `void activemq::core::ActiveMQConsumer::setRedeliveryPolicy ( RedeliveryPolicy * policy ) [inline]`

Sets the **RedeliveryPolicy** (p. 3267) this Consumer should use when a rollback is performed on a transacted Consumer.

The Consumer takes ownership of the passed pointer. The Consumer's redelivery policy can never be null, a call to this method with a NULL pointer is ignored.

#### Parameters

<i>policy</i>	Pointer to a Redelivery Policy object that his Consumer will use.
---------------	---

**6.27.2.30** `void activemq::core::ActiveMQConsumer::setSynchronizationRegistered ( bool value ) [inline]`

Sets the **Synchronization** (p. 3826) Registered state of this consumer.

#### Parameters

<i>value</i>	- true if registered false otherwise.
--------------	---------------------------------------

**6.27.2.31** `virtual void activemq::core::ActiveMQConsumer::start ( ) [virtual]`

Starts the Consumer if not already started and not closed.

A consumer will no deliver messages until started.

**6.27.2.32** `virtual void activemq::core::ActiveMQConsumer::stop ( ) [virtual]`

Stops a Consumer, the Consumer will not deliver any messages that are dispatched to it until it is started again.

A Closed Consumer is also a stopped consumer.

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConsumer.h**

## 6.28 activemq::library::ActiveMQCPP Class Reference

```
#include <src/main/activemq/library/ActiveMQCPP.h>
```

## Public Member Functions

- virtual `~ActiveMQCPP()`

## Static Public Member Functions

- static void **initializeLibrary** ()  
*Initialize the ActiveMQ-CPP Library constructs, this method will init all the internal Registry objects and initialize the Decaf library.*
- static void **initializeLibrary** (int argc, char \*\*argv)  
*Initialize the ActiveMQ-CPP Library constructs, this method will initialize all the internal Registry objects and initialize the Decaf library.*
- static void **shutdownLibrary** ()  
*Shutdown the ActiveMQ-CPP Library, freeing any resources that could not be freed up to this point.*

## Protected Member Functions

- **ActiveMQCPP** ()
- **ActiveMQCPP** (const **ActiveMQCPP** &)
- **ActiveMQCPP** & **operator=** (const **ActiveMQCPP** &)

### 6.28.1 Constructor & Destructor Documentation

6.28.1.1 `activemq::library::ActiveMQCPP::ActiveMQCPP ( )` [`inline`, `protected`]

6.28.1.2 `activemq::library::ActiveMQCPP::ActiveMQCPP ( const ActiveMQCPP & )`  
[`protected`]

6.28.1.3 `virtual activemq::library::ActiveMQCPP::~~ActiveMQCPP ( )` [`inline`, `virtual`]

### 6.28.2 Member Function Documentation

6.28.2.1 `static void activemq::library::ActiveMQCPP::initializeLibrary ( )` [`static`]

Initialize the ActiveMQ-CPP Library constructs, this method will init all the internal Registry objects and initialize the Decaf library.

## Exceptions

<code>runtime_error</code>	if an error occurs while initializing this library.
----------------------------	---

**6.28.2.2** `static void activemq::library::ActiveMQCPP::initializeLibrary ( int argc, char ** argv )`  
`[static]`

Initialize the ActiveMQ-CPP Library constructs, this method will initialize all the internal Registry objects and initialize the Decaf library.

This method takes the args passed to the main method of process for use is setting system properties and configuring the ActiveMQ-CPP Library.

#### Parameters

<i>argc</i>	- the count of arguments passed to this Process.
<i>argv</i>	- the array of string arguments passed to this process.

#### Exceptions

<i>runtime_error</i>	if an error occurs while initializing this library.
----------------------	---

**6.28.2.3** `ActiveMQCPP& activemq::library::ActiveMQCPP::operator= ( const ActiveMQCPP & )`  
`[protected]`

**6.28.2.4** `static void activemq::library::ActiveMQCPP::shutdownLibrary ( )`  
`[static]`

Shutdown the ActiveMQ-CPP Library, freeing any resources that could not be freed up to this point.

All the user created ActiveMQ-CPP objects and Decaf Library objects should have been destroyed by the time this method is called.

The documentation for this class was generated from the following file:

- `src/main/activemq/library/ActiveMQCPP.h`

## 6.29 activemq::commands::ActiveMQDestination Class Reference

```
#include <src/main/activemq/commands/ActiveMQDestination.h>
```

Inheritance diagram for `activemq::commands::ActiveMQDestination`:

#### Data Structures

- struct `DestinationFilter`

#### Public Member Functions

- `ActiveMQDestination ( )`



- **ActiveMQDestination** (const std::string &physicalName)
- virtual ~**ActiveMQDestination** ()
- virtual **ActiveMQDestination** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual unsigned char **getDataStructureType** () const  
*Get the **DataStructure** (p. 1713) Type as defined in CommandTypes.h.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual const std::string & **getPhysicalName** () const  
*Fetch this destination's physical name.*
- virtual std::string & **getPhysicalName** ()
- virtual void **setPhysicalName** (const std::string &physicalName)  
*Set this destination's physical name.*
- virtual bool **isAdvisory** () const
- virtual void **setAdvisory** (bool advisory)
- virtual bool **isConsumerAdvisory** () const
- virtual bool **isProducerAdvisory** () const
- virtual bool **isConnectionAdvisory** () const
- virtual bool **isExclusive** () const
- virtual void **setExclusive** (bool exclusive)
- virtual bool **isOrdered** () const
- virtual void **setOrdered** (bool ordered)
- virtual std::string **getOrderedTarget** () const
- virtual void **setOrderedTarget** (const std::string &orderedTarget)
- virtual **cms::Destination::DestinationType** **getDestinationType** () const =0  
*Returns the Type of Destination that this object represents.*
- virtual bool **isTemporary** () const  
*Returns true if a temporary Destination.*
- virtual bool **isTopic** () const  
*Returns true if a Topic Destination.*

- virtual bool **isQueue** () const  
*Returns true if a Queue Destination.*
- virtual bool **isComposite** () const  
*Returns true if this destination represents a collection of destinations; allowing a set of destinations to be published to or subscribed from in one CMS operation.*
- virtual bool **isWildcard** () const
- const **activemq::util::ActiveMQProperties** & **getOptions** () const
- virtual const **cms::Destination** \* **getCMSDestination** () const

### Static Public Member Functions

- static std::string **createTemporaryName** (const std::string &clientId)  
*Create a temporary name from the clientId.*
- static std::string **getClientId** (const **ActiveMQDestination** \*destination)  
*From a temporary destination find the clientId of the Connection that created it.*
- static **Pointer**< **ActiveMQDestination** > **createDestination** (int type, const std::string &name)  
*Creates a Destination given the String Name to use and a Type.*

### Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQDESTINATION** = 0

### Protected Attributes

- bool **exclusive**
- bool **ordered**
- bool **advisory**
- std::string **orderedTarget**
- std::string **physicalName**
- **util::ActiveMQProperties** **options**

### Static Protected Attributes

- static const std::string **ADVISORY\_PREFIX**  
*prefix for Advisory message destinations*
- static const std::string **CONSUMER\_ADVISORY\_PREFIX**  
*prefix for consumer advisory destinations*

- static const std::string **PRODUCER\_ADVISORY\_PREFIX**  
*prefix for producer advisory destinations*
- static const std::string **CONNECTION\_ADVISORY\_PREFIX**  
*prefix for connection advisory destinations*
- static const std::string **DEFAULT\_ORDERED\_TARGET**  
*The default target for ordered destinations.*
- static const std::string **TEMP\_PREFIX**
- static const std::string **TEMP\_POSTFIX**
- static const std::string **COMPOSITE\_SEPARATOR**
- static const std::string **QUEUE\_QUALIFIED\_PREFIX**
- static const std::string **TOPIC\_QUALIFIED\_PREFIX**
- static const std::string **TEMP\_QUEUE\_QUALIFIED\_PREFIX**
- static const std::string **TEMP\_TOPIC\_QUALIFIED\_PREFIX**

### 6.29.1 Constructor & Destructor Documentation

6.29.1.1 `activemq::commands::ActiveMQDestination::ActiveMQDestination ( )`

6.29.1.2 `activemq::commands::ActiveMQDestination::ActiveMQDestination ( const std::string & physicalName )`

6.29.1.3 `virtual activemq::commands::ActiveMQDestination::~~ActiveMQDestination ( )`  
[inline, virtual]

### 6.29.2 Member Function Documentation

6.29.2.1 `virtual ActiveMQDestination* activemq::commands::ActiveMQDestination::cloneDataStructure ( )`  
`const` [inline, virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1714).

Reimplemented in `activemq::commands::ActiveMQQueue` (p. 481), `activemq::commands::ActiveMQTempDestination` (p. 580), `activemq::commands::ActiveMQTempQueue` (p. 608), `activemq::commands::ActiveMQTempTopic` (p. 638), and `activemq::commands::ActiveMQTopic` (p. 698).

**6.29.2.2** `virtual void activemq::commands::ActiveMQDestination::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

*src* - Source Object

Implements `activemq::commands::DataStructure` (p. 1715).

Reimplemented in `activemq::commands::ActiveMQQueue` (p. 481), `activemq::commands::ActiveMQTempQueue` (p. 580), `activemq::commands::ActiveMQTempQueue` (p. 609), `activemq::commands::ActiveMQTempTopic` (p. 639), and `activemq::commands::ActiveMQTopic` (p. 699).

Referenced by `activemq::commands::ActiveMQTempDestination::copyDataStructure()`.

**6.29.2.3** `static Pointer<ActiveMQDestination> activemq::commands::ActiveMQDestination::createDestination ( int type, const std::string & name ) [static]`

Creates a Destination given the String Name to use and a Type.

#### Parameters

<i>type</i>	- The Type of Destination to Create
<i>name</i>	- The Name to use in the creation of the Destination

#### Returns

Pointer to a new `ActiveMQDestination` (p. 312) instance.

**6.29.2.4** `static std::string activemq::commands::ActiveMQDestination::createTemporaryName ( const std::string & clientId ) [inline, static]`

Create a temporary name from the clientId.

#### Parameters

<i>clientId</i>	
-----------------	--

#### Returns

**6.29.2.5** `virtual bool activemq::commands::ActiveMQDestination::equals ( const DataStructure * value ) const` [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1716).

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 481), **activemq::commands::ActiveMQTempDestination** (p. 581), **activemq::commands::ActiveMQTempQueue** (p. 609), **activemq::commands::ActiveMQTempTopic** (p. 639), and **activemq::commands::ActiveMQTopic** (p. 699).

Referenced by **activemq::commands::ActiveMQTopic::equals()**, and **activemq::commands::ActiveMQTempDestination::equals()**.

**6.29.2.6** `static std::string activemq::commands::ActiveMQDestination::getClientId ( const ActiveMQDestination * destination )` [static]

From a temporary destination find the clientId of the Connection that created it.

#### Parameters

<i>destination</i>	
--------------------	--

#### Returns

the clientId or null if not a temporary destination

**6.29.2.7** `virtual const cms::Destination* activemq::commands::ActiveMQDestination::getCMSDestination ( ) const` [inline, virtual]

#### Returns

the **cms::Destination** (p. 1776) interface pointer that the objects that derive from this class implement.

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 482), **activemq::commands::ActiveMQTempQueue** (p. 609), **activemq::commands::ActiveMQTempTopic** (p. 639), and **activemq::commands::ActiveMQTopic** (p. 699).

**6.29.2.8** `virtual unsigned char activemq::commands::ActiveMQDestination::getDataStructureType ( ) const` [virtual]

Get the **DataStructure** (p. 1713) Type as defined in CommandTypes.h.

**Returns**

The type of the data structure

Implements `activemq::commands::DataStructure` (p. 1717).

Reimplemented in `activemq::commands::ActiveMQQueue` (p. 482), `activemq::commands::ActiveMQTempQueue` (p. 581), `activemq::commands::ActiveMQTempQueue` (p. 610), `activemq::commands::ActiveMQTempTopic` (p. 640), and `activemq::commands::ActiveMQTopic` (p. 700).

**6.29.2.9 virtual cms::Destination::DestinationType**

```
activemq::commands::ActiveMQDestination::getDestinationType ( ) const [pure
virtual]
```

Returns the Type of Destination that this object represents.

**Returns**

int type qualifier.

Implemented in `activemq::commands::ActiveMQQueue` (p. 482), `activemq::commands::ActiveMQTempQueue` (p. 610), `activemq::commands::ActiveMQTempTopic` (p. 640), and `activemq::commands::ActiveMQTopic` (p. 700).

**6.29.2.10 const activemq::util::ActiveMQProperties&**

```
activemq::commands::ActiveMQDestination::getOptions ( ) const [inline]
```

**Returns**

a reference (const) to the options properties for this Destination.

**6.29.2.11 virtual std::string activemq::commands::ActiveMQDestination::getOrderedTarget ( )**

```
const [inline, virtual]
```

**Returns**

Returns the orderedTarget.

**6.29.2.12 virtual const std::string& activemq::commands::ActiveMQDestination::getPhysicalName**

```
( ) const [inline, virtual]
```

Fetch this destination's physical name.

**Returns**

const string containing the name

**6.29.2.13** `virtual std::string& activemq::commands::ActiveMQDestination::getPhysicalName ( )`  
[inline, virtual]

**6.29.2.14** `virtual bool activemq::commands::ActiveMQDestination::isAdvisory ( ) const`  
[inline, virtual]

#### Returns

Returns the advisory.

**6.29.2.15** `virtual bool activemq::commands::ActiveMQDestination::isComposite ( ) const`  
[inline, virtual]

Returns true if this destination represents a collection of destinations; allowing a set of destinations to be published to or subscribed from in one CMS operation.

#### Returns

true if this destination represents a collection of child destinations.

**6.29.2.16** `virtual bool activemq::commands::ActiveMQDestination::isConnectionAdvisory ( )`  
`const` [inline, virtual]

#### Returns

true if this is a destination for Connection advisories

**6.29.2.17** `virtual bool activemq::commands::ActiveMQDestination::isConsumerAdvisory ( )`  
`const` [inline, virtual]

#### Returns

true if this is a destination for Consumer advisories

**6.29.2.18** `virtual bool activemq::commands::ActiveMQDestination::isExclusive ( ) const`  
[inline, virtual]

#### Returns

Returns the exclusive.

**6.29.2.19** `virtual bool activemq::commands::ActiveMQDestination::isOrdered ( ) const`  
[inline, virtual]

#### Returns

Returns the ordered.

**6.29.2.20** `virtual bool activemq::commands::ActiveMQDestination::isProducerAdvisory ( ) const`  
[inline, virtual]

#### Returns

true if this is a destination for Producer advisories

**6.29.2.21** `virtual bool activemq::commands::ActiveMQDestination::isQueue ( ) const`  
[inline, virtual]

Returns true if a Queue Destination.

#### Returns

true/false

**6.29.2.22** `virtual bool activemq::commands::ActiveMQDestination::isTemporary ( ) const`  
[inline, virtual]

Returns true if a temporary Destination.

#### Returns

true/false

References cms::Destination::TEMPORARY\_QUEUE, and cms::Destination::TEMPORARY\_TOPIC.

**6.29.2.23** `virtual bool activemq::commands::ActiveMQDestination::isTopic ( ) const`  
[inline, virtual]

Returns true if a Topic Destination.

#### Returns

true/false

References cms::Destination::TEMPORARY\_TOPIC, and cms::Destination::TOPIC.

**6.29.2.24** `virtual bool activemq::commands::ActiveMQDestination::isWildcard ( ) const`  
[inline, virtual]

#### Returns

true if the destination matches multiple possible destinations



**6.29.2.25** `virtual void activemq::commands::ActiveMQDestination::setAdvisory ( bool advisory ) [inline, virtual]`

#### Parameters

<i>advisory</i>	The advisory to set.
-----------------	----------------------

**6.29.2.26** `virtual void activemq::commands::ActiveMQDestination::setExclusive ( bool exclusive ) [inline, virtual]`

#### Parameters

<i>exclusive</i>	The exclusive to set.
------------------	-----------------------

**6.29.2.27** `virtual void activemq::commands::ActiveMQDestination::setOrdered ( bool ordered ) [inline, virtual]`

#### Parameters

<i>ordered</i>	The ordered to set.
----------------	---------------------

**6.29.2.28** `virtual void activemq::commands::ActiveMQDestination::setOrderedTarget ( const std::string & orderedTarget ) [inline, virtual]`

#### Parameters

<i>orderedTarget</i>	The orderedTarget to set.
----------------------	---------------------------

**6.29.2.29** `virtual void activemq::commands::ActiveMQDestination::setPhysicalName ( const std::string & physicalName ) [virtual]`

Set this destination's physical name.

#### Returns

const string containing the name

**6.29.2.30** `virtual std::string activemq::commands::ActiveMQDestination::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p. 841).

Reimplemented in `activemq::commands::ActiveMQQueue` (p. 483), `activemq::commands::ActiveMQTempQueue` (p. 582), `activemq::commands::ActiveMQTempQueue` (p. 611), `activemq::commands::ActiveMQTempTopic` (p. 641), and `activemq::commands::ActiveMQTopic` (p. 701).

### 6.29.3 Field Documentation

**6.29.3.1** `bool activemq::commands::ActiveMQDestination::advisory`  
[protected]

**6.29.3.2** `const std::string activemq::commands::ActiveMQDestination::ADVISORY_PREFIX` [static, protected]

prefix for Advisory message destinations

**6.29.3.3** `const std::string activemq::commands::ActiveMQDestination::COMPOSITE_SEPARATOR` [static, protected]

**6.29.3.4** `const std::string activemq::commands::ActiveMQDestination::CONNECTION_ADVISORY_PREFIX` [static, protected]

prefix for connection advisory destinations

**6.29.3.5** `const std::string activemq::commands::ActiveMQDestination::CONSUMER_ADVISORY_PREFIX` [static, protected]

prefix for consumer advisory destinations

**6.29.3.6** `const std::string activemq::commands::ActiveMQDestination::DEFAULT_ORDERED_TARGET` [static, protected]

The default target for ordered destinations.

- 6.29.3.7 **bool** `activemq::commands::ActiveMQDestination::exclusive`  
[protected]
- 6.29.3.8 **const unsigned char** `activemq::commands::ActiveMQDestination::ID_ -  
ACTIVEMQDESTINATION = 0` [static]
- 6.29.3.9 **util::ActiveMQProperties** `activemq::commands::ActiveMQDestination::options`  
[protected]
- 6.29.3.10 **bool** `activemq::commands::ActiveMQDestination::ordered`  
[protected]
- 6.29.3.11 **std::string** `activemq::commands::ActiveMQDestination::orderedTarget`  
[protected]
- 6.29.3.12 **std::string** `activemq::commands::ActiveMQDestination::physicalName`  
[protected]
- 6.29.3.13 **const std::string** `activemq::commands::ActiveMQDestination::PRODUCER_ -  
ADVISORY_PREFIX` [static, protected]

prefix for producer advisory destinations

- 6.29.3.14 **const std::string** `activemq::commands::ActiveMQDestination::QUEUE_ -  
QUALIFIED_PREFIX` [static, protected]
- 6.29.3.15 **const std::string** `activemq::commands::ActiveMQDestination::TEMP_ -  
POSTFIX` [static, protected]
- 6.29.3.16 **const std::string** `activemq::commands::ActiveMQDestination::TEMP_ -  
PREFIX` [static, protected]
- 6.29.3.17 **const std::string** `activemq::commands::ActiveMQDestination::TEMP_ -  
QUEUE_QUALIFIED_PREFIX` [static,  
protected]
- 6.29.3.18 **const std::string** `activemq::commands::ActiveMQDestination::TEMP_ -  
TOPIC_QUALIFIED_PREFIX` [static,  
protected]
- 6.29.3.19 **const std::string** `activemq::commands::ActiveMQDestination::TOPIC_ -  
QUALIFIED_PREFIX` [static, protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQDestination.h`

## 6.30 activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 324).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestina
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller`:

### Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.30.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 324).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

### 6.30 ac-

activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller

Class Reference

325

#### 6.30.2 Constructor & Destructor Documentation

6.30.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller ( ) [inline]`

6.30.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller ( ) [inline, virtual]`

#### 6.30.3 Member Function Documentation

6.30.3.1 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

##### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

##### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1675).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller` (p. 488), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 584), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 613), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 643), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller` (p. 703).

6.30.3.2 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

##### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1683).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller` (p. 489), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 584), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 613), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 643), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller` (p. 703).

```
6.30.3.3  virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1690).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller` (p. 489), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 585), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 614), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 644), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller` (p. 704).

```
6.30.3.4  virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### 6.30 ac-

**activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller**

#### Class Reference

327

##### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

##### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 490), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 585), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 614), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 644), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller** (p. 704).

**6.30.3.5** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

##### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

##### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 490), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 586), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 615), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 645), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller** (p. 705).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h`

## 6.31 activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 328).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestina
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller`:

### Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.31.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 328).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module



- 6.31.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller ( ) [inline]`
- 6.31.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller ( ) [inline, virtual]`

### 6.31.3 Member Function Documentation

- 6.31.3.1 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1675).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 493), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 588), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 617), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 651), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 711).

- 6.31.3.2 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1683).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 493), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 588), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 617), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 652), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 712).

```
6.31.3.3  virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1690).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 494), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 589), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 618), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 652), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 712).

```
6.31.3.4  virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### 6.31 activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller

#### Class Reference 331

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 494), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 589), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 618), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 653), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 713).

**6.31.3.5** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 495), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 590), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 619), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 653), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 713).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h`

## 6.32 activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 332).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestina
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller`:

### Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.32.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 332).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

- 6.32.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller ( ) [inline]`
- 6.32.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller ( ) [inline, virtual]`

### 6.32.3 Member Function Documentation

- 6.32.3.1 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1675).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` (p. 497), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 592), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 621), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 647), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller` (p. 707).

- 6.32.3.2 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller** (p. 497), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 592), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 622), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 647), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller** (p. 708).

```
6.32.3.3  virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller** (p. 498), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 593), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 622), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 648), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller** (p. 708).

```
6.32.3.4  virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

## 6.32 ac-

**activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller**

### Class Reference

335

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller** (p. 498), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 593), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 623), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 648), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller** (p. 709).

**6.32.3.5** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller** (p. 499), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 594), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 623), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 649), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller** (p. 709).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h`

### 6.33 activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 336).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestina
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller:

#### Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

#### 6.33.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 336).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



### 6.33 ac-

activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller

Class Reference

337

#### 6.33.2 Constructor & Destructor Documentation

6.33.2.1 **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller**  
( ) [inline]

6.33.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller**  
( ) [inline, virtual]

#### 6.33.3 Member Function Documentation

6.33.3.1 **virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::looseMarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
\* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

##### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

##### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller** (p. 501), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 596), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 625), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 655), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller** (p. 716).

6.33.3.2 **virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::looseUnmarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
\* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

##### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1683).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller` (p. 502), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 596), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 626), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 656), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 716).

```
6.33.3.3 virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1690).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller` (p. 502), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 597), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 626), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 656), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 717).

```
6.33.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### 6.33 ac-

**activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller**

#### Class Reference

339

##### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

##### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller** (p. 503), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 597), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 627), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 657), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller** (p. 717).

**6.33.3.5** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

##### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

##### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller** (p. 503), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 598), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 627), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 657), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller** (p. 718).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h`

## 6.34 activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 340).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestina
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller`:

### Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.34.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 340).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

- 6.34.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller ( ) [inline]`
- 6.34.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller ( ) [inline, virtual]`

### 6.34.3 Member Function Documentation

- 6.34.3.1 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1675).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 505), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 600), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 630), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 660), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 724).

- 6.34.3.2 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1683).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 506), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 600), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 630), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 660), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 725).

```
6.34.3.3  virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1690).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 506), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 601), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 631), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 661), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 725).

```
6.34.3.4  virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### 6.34 ac-

**activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller**

#### Class Reference

343

##### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

##### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller** (p. 507), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 601), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 631), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 661), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller** (p. 726).

**6.34.3.5** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

##### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

##### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller** (p. 507), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 602), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 632), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 662), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller** (p. 726).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h`

## 6.35 activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 344).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQDestina
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller`:

### Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.35.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 344).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module



- 6.35.2.1 `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller ( ) [inline]`
- 6.35.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller ( ) [inline, virtual]`

### 6.35.3 Member Function Documentation

- 6.35.3.1 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1675).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller` (p. 510), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 604), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller` (p. 634), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller` (p. 664), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller` (p. 720).

- 6.35.3.2 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller** (p. 510), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 604), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 634), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 664), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller** (p. 720).

```
6.35.3.3  virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller** (p. 511), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 605), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 635), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 665), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller** (p. 721).

```
6.35.3.4  virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### 6.35 ac-

**activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller**

#### Class Reference

347

##### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

##### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller** (p. 511), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 605), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 635), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 665), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller** (p. 721).

**6.35.3.5** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

##### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

##### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller** (p. 512), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 606), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 636), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 666), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller** (p. 722).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h`

## 6.36 activemq::exceptions::ActiveMQException Class Reference

```
#include <src/main/activemq/exceptions/ActiveMQException.h>
```

Inheritance diagram for activemq::exceptions::ActiveMQException:

### Public Member Functions

- **ActiveMQException** () throw ()  
*Default Constructor.*
- **ActiveMQException** (const **ActiveMQException** &ex) throw ()  
*Copy Constructor.*
- **ActiveMQException** (const **decaf::lang::Exception** &ex) throw ()  
*Copy Constructor.*
- **ActiveMQException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual ~**ActiveMQException** () throw ()
- virtual **ActiveMQException** \* **clone** () const  
*Clones this exception.*
- virtual **cms::CMSException** **convertToCMSException** () const  
*Converts this exception to a new CMSException.*

### 6.36.1 Constructor & Destructor Documentation

#### 6.36.1.1 activemq::exceptions::ActiveMQException::ActiveMQException ( ) throw ()

Default Constructor.

#### 6.36.1.2 activemq::exceptions::ActiveMQException::ActiveMQException ( const **ActiveMQException** & ex ) throw ()

Copy Constructor.

### Parameters

<i>ex</i>	The Exception whose internal data is copied into this instance.
-----------	---

**6.36.1.3** `activemq::exceptions::ActiveMQException::ActiveMQException ( const decaf::lang::Exception & ex ) throw ()`

Copy Constructor.

#### Parameters

<i>ex</i>	The Exception whose internal data is copied into this instance.
-----------	---

**6.36.1.4** `activemq::exceptions::ActiveMQException::ActiveMQException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()`

Constructor - Initializes the file name and line number where this message occurred.  
Sets the message to report, using an optional list of arguments to parse into the message.

#### Parameters

<i>file</i>	The file name where exception occurs.
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report.
<i>...</i>	The list of primitives that are formatted into the message.

**6.36.1.5** `virtual activemq::exceptions::ActiveMQException::~~ActiveMQException ( ) throw ()`  
[virtual]

### 6.36.2 Member Function Documentation

**6.36.2.1** `virtual ActiveMQException* activemq::exceptions::ActiveMQException::clone ( ) const` [virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

Copy of this Exception object

Reimplemented from **decaf::lang::Exception** (p. 1889).

Reimplemented in **activemq::exceptions::BrokerException** (p. 874).

**6.36.2.2** `virtual cms::CMSException activemq::exceptions::ActiveMQException::convertToCMSException ( ) const` [virtual]

Converts this exception to a new CMSException.

**Returns**

a CMSEException with the data from this exception

The documentation for this class was generated from the following file:

- src/main/activemq/exceptions/**ActiveMQException.h**

**6.37 activemq::commands::ActiveMQMapMessage Class Reference**

```
#include <src/main/activemq/commands/ActiveMQMapMessage.h>
```

Inheritance diagram for activemq::commands::ActiveMQMapMessage:

**Public Member Functions**

- **ActiveMQMapMessage ()**
- virtual **~ActiveMQMapMessage ()**
- virtual unsigned char **getDataStructureType ()** const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual bool **isMarshalAware ()** const  
*Determine if this object is aware of marshaling and should have its before and after marshaling methods called.*
- virtual **ActiveMQMapMessage \* cloneDataStructure ()** const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure (const DataStructure \*src)**  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual void **beforeMarshal (wireformat::WireFormat \*wireFormat) throw ( decaf::io::IOException )**  
*Perform any processing needed before an marshal.*
- virtual std::string **toString ()** const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals (const DataStructure \*value)** const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*

- virtual void **clearBody** () throw ( cms::CMSException )  
*Clears out the body of the message.*
- virtual cms::MapMessage \* **clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*
- virtual std::vector< std::string > **getMapNames** () const throw ( cms::CMSException )  
*Returns an Enumeration of all the names in the MapMessage object.*
- virtual bool **itemExists** (const std::string &name) const throw ( cms::CMSException )  
*Indicates whether an item exists in this MapMessage object.*
- virtual bool **getBoolean** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSException )  
*Returns the Boolean value of the Specified name.*
- virtual void **setBoolean** (const std::string &name, bool value) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Sets a boolean value with the specified name into the Map.*
- virtual unsigned char **getByte** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSException )  
*Returns the Byte value of the Specified name.*
- virtual void **setByte** (const std::string &name, unsigned char value) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Sets a Byte value with the specified name into the Map.*
- virtual std::vector< unsigned char > **getBytes** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSException )  
*Returns the Bytes value of the Specified name.*
- virtual void **setBytes** (const std::string &name, const std::vector< unsigned char > &value) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Sets a Bytes value with the specified name into the Map.*
- virtual char **getChar** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSException )  
*Returns the Char value of the Specified name.*
- virtual void **setChar** (const std::string &name, char value) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Sets a Char value with the specified name into the Map.*

- virtual double **getDouble** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Double value of the Specified name.*
- virtual void **setDouble** (const std::string &name, double value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Double value with the specified name into the Map.*
- virtual float **getFloat** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Float value of the Specified name.*
- virtual void **setFloat** (const std::string &name, float value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Float value with the specified name into the Map.*
- virtual int **getInt** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Int value of the Specified name.*
- virtual void **setInt** (const std::string &name, int value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Int value with the specified name into the Map.*
- virtual long long **getLong** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Long value of the Specified name.*
- virtual void **setLong** (const std::string &name, long long value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Long value with the specified name into the Map.*
- virtual short **getShort** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Short value of the Specified name.*
- virtual void **setShort** (const std::string &name, short value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Short value with the specified name into the Map.*
- virtual std::string **getString** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the String value of the Specified name.*
- virtual void **setString** (const std::string &name, const std::string &value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a String value with the specified name into the Map.*



## Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQMAPMESSAGE** = 25

## Protected Member Functions

- **util::PrimitiveMap & getMap ()** throw ( decaf::lang::exceptions::NullPointerException )  
*Fetches a reference to this objects PrimitiveMap, if one needs to be created or unmarshaled, this will perform the correct steps.*
- const **util::PrimitiveMap & getMap ()** const throw ( decaf::lang::exceptions::NullPointerException )
- virtual void **checkMapIsUnmarshalled ()** const throw ( decaf::lang::exceptions::NullPointerException )  
*Performs the unmarshal on the Map if needed, otherwise just returns.*

### 6.37.1 Constructor & Destructor Documentation

6.37.1.1 **activemq::commands::ActiveMQMapMessage::ActiveMQMapMessage ( )**

6.37.1.2 **virtual activemq::commands::ActiveMQMapMessage::~~ActiveMQMapMessage ( )**  
 [virtual]

### 6.37.2 Member Function Documentation

6.37.2.1 **virtual void activemq::commands::ActiveMQMapMessage::beforeMarshal ( wireformat::WireFormat \* wireFormat )** throw ( decaf::io::IOException )  
 [virtual]

Perform any processing needed before an marshal.

#### Parameters

<i>wireFormat</i>	- the OpenWireFormat object in use.
-------------------	-------------------------------------

Implements **activemq::wireformat::MarshalAware** (p. 2565).

6.37.2.2 **virtual void activemq::commands::ActiveMQMapMessage::checkMapIsUnmarshalled ( )** const throw ( decaf::lang::exceptions::NullPointerException )  
 [protected, virtual]

Performs the unmarshal on the Map if needed, otherwise just returns.

**6.37.2.3** `virtual void activemq::commands::ActiveMQMapMessage::clearBody ( ) throw ( cms::CMSException ) [virtual]`

Clears out the body of the message.

This does not clear the headers or properties.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 422).

**6.37.2.4** `virtual cms::MapMessage* activemq::commands::ActiveMQMapMessage::clone ( ) const [inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

#### Returns

new copy of this message

Implements `cms::Message` (p. 2620).

**6.37.2.5** `virtual ActiveMQMapMessage* activemq::commands::ActiveMQMapMessage::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2601).

**6.37.2.6** `virtual void activemq::commands::ActiveMQMapMessage::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2602).

**6.37.2.7** `virtual bool activemq::commands::ActiveMQMapMessage::equals ( const DataStructure * value ) const` [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 423).

**6.37.2.8** `virtual bool activemq::commands::ActiveMQMapMessage::getBoolean ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSEException )` [virtual]

Returns the Boolean value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

#### Exceptions

<i>CMSEException</i>	- if the operation fails due to an internal error.
<i>MessageFormatException</i>	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 2554).

**6.37.2.9** `virtual unsigned char activemq::commands::ActiveMQMapMessage::getByte ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSEException )` [virtual]

Returns the Byte value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

#### Exceptions

<i>CMSEException</i>	- if the operation fails due to an internal error.
<i>MessageFormatException</i>	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 2554).

**6.37.2.10** `virtual std::vector<unsigned char> activemq::commands::ActiveQMapMessage::getBytes ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSException ) [virtual]`

Returns the Bytes value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

#### Exceptions

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageFormatException</i>	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 2555).

**6.37.2.11** `virtual char activemq::commands::ActiveQMapMessage::getChar ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSException ) [virtual]`

Returns the Char value of the Specified name.

#### Parameters

<i>name</i>	name of the value to fetch from the map
-------------	---

#### Exceptions

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageFormatException</i>	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 2555).

**6.37.2.12** `virtual unsigned char activemq::commands::ActiveQMapMessage::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1713) type copy.

Reimplemented from **activemq::commands::Message** (p. 2604).

**6.37.2.13** virtual double activemq::commands::ActiveMQMapMessage::getDouble ( const std::string & *name* ) const throw ( cms::MessageFormatException, cms::CMSException ) [virtual]

Returns the Double value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

#### Exceptions

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageFormatException</i>	- if this type conversion is invalid.

Implements cms::MapMessage (p. 2555).

**6.37.2.14** virtual float activemq::commands::ActiveMQMapMessage::getFloat ( const std::string & *name* ) const throw ( cms::MessageFormatException, cms::CMSException ) [virtual]

Returns the Float value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

#### Exceptions

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageFormatException</i>	- if this type conversion is invalid.

Implements cms::MapMessage (p. 2556).

**6.37.2.15** virtual int activemq::commands::ActiveMQMapMessage::getInt ( const std::string & *name* ) const throw ( cms::MessageFormatException, cms::CMSException ) [virtual]

Returns the Int value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

#### Exceptions

<i>CMSException</i>	- if the operation fails due to an internal error.
---------------------	--

<i>MessageFormatException</i>	- if this type conversion is invalid.
-------------------------------	---------------------------------------

Implements **cms::MapMessage** (p. 2556).

**6.37.2.16** `virtual long long activemq::commands::ActiveMQMapMessage::getLong ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSEException ) [virtual]`

Returns the Long value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

#### Exceptions

<i>CMSEException</i>	- if the operation fails due to an internal error.
<i>MessageFormatException</i>	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 2557).

**6.37.2.17** `util::PrimitiveMap& activemq::commands::ActiveMQMapMessage::getMap ( ) throw ( decaf::lang::exceptions::NullPointerException ) [protected]`

Fetches a reference to this objects PrimitiveMap, if one needs to be created or unmarshaled, this will perform the correct steps.

#### Returns

reference to a PrimitiveMap.

**6.37.2.18** `const util::PrimitiveMap& activemq::commands::ActiveMQMapMessage::getMap ( ) const throw ( decaf::lang::exceptions::NullPointerException ) [protected]`

**6.37.2.19** `virtual std::vector< std::string > activemq::commands::ActiveMQMapMessage::getMapNames ( ) const throw ( cms::CMSEException ) [virtual]`

Returns an Enumeration of all the names in the MapMessage object.

#### Returns

STL Vector of String values, each of which is the name of an item in the MapMessage

**Exceptions**

<i>CMSException</i>	- if the operation fails due to an internal error.
---------------------	--

Implements **cms::MapMessage** (p. 2557).

**6.37.2.20** `virtual short activemq::commands::ActiveMQMapMessage::getShort ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSException ) [virtual]`

Returns the Short value of the Specified name.

**Parameters**

<i>name</i>	Name of the value to fetch from the map
-------------	---

**Exceptions**

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageFormatException</i>	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 2557).

**6.37.2.21** `virtual std::string activemq::commands::ActiveMQMapMessage::getString ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSException ) [virtual]`

Returns the String value of the Specified name.

**Parameters**

<i>name</i>	Name of the value to fetch from the map
-------------	---

**Exceptions**

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageFormatException</i>	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 2558).

**6.37.2.22** `virtual bool activemq::commands::ActiveMQMapMessage::isMarshalAware ( ) const [inline, virtual]`

Determine if this object is aware of marshaling and should have its before and after marshaling methods called.

Defaults to false.

**Returns**

true if aware of marshaling

Reimplemented from **activemq::commands::Message** (p. 2607).

**6.37.2.23** `virtual bool activemq::commands::ActiveMQMapMessage::itemExists ( const std::string & name ) const throw ( cms::CMSException )` [virtual]

Indicates whether an item exists in this MapMessage object.

**Parameters**

<i>name</i>	String name of the Object in question
-------------	---------------------------------------

**Returns**

boolean value indicating if the name is in the map

**Exceptions**

<i>CMSException</i>	- if the operation fails due to an internal error.
---------------------	--

Implements **cms::MapMessage** (p. 2558).

**6.37.2.24** `virtual void activemq::commands::ActiveMQMapMessage::setBoolean ( const std::string & name, bool value ) throw ( cms::MessageNotWriteableException, cms::CMSException )` [virtual]

Sets a boolean value with the specified name into the Map.

**Parameters**

<i>name</i>	the name of the boolean
<i>value</i>	the boolean value to set in the Map

**Exceptions**

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i>	- if the <b>Message</b> (p. 2596) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2559).

**6.37.2.25** `virtual void activemq::commands::ActiveMQMapMessage::setByte ( const std::string & name, unsigned char value ) throw ( cms::MessageNotWriteableException, cms::CMSException )` [virtual]

Sets a Byte value with the specified name into the Map.



**Parameters**

<i>name</i>	the name of the Byte
<i>value</i>	the Byte value to set in the Map

**Exceptions**

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i>	- if the <b>Message</b> (p. 2596) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2559).

**6.37.2.26** `virtual void activemq::commands::ActiveMQMapMessage::setBytes ( const std::string & name, const std::vector< unsigned char > & value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Sets a Bytes value with the specified name into the Map.

**Parameters**

<i>name</i>	The name of the Bytes
<i>value</i>	The Bytes value to set in the Map

**Exceptions**

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i>	- if the <b>Message</b> (p. 2596) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2559).

**6.37.2.27** `virtual void activemq::commands::ActiveMQMapMessage::setChar ( const std::string & name, char value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Sets a Char value with the specified name into the Map.

**Parameters**

<i>name</i>	the name of the Char
<i>value</i>	the Char value to set in the Map

**Exceptions**

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i>	- if the <b>Message</b> (p. 2596) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2560).

```
6.37.2.28  virtual void activemq::commands::ActiveMQMapMessage::setDouble
            ( const std::string & name, double value ) throw (
              cms::MessageNotWriteableException, cms::CMSEException )
            [virtual]
```

Sets a Double value with the specified name into the Map.

#### Parameters

<i>name</i>	The name of the Double
<i>value</i>	The Double value to set in the Map

#### Exceptions

<i>CMSEException</i>	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i>	- if the <b>Message</b> (p. 2596) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2560).

```
6.37.2.29  virtual void activemq::commands::ActiveMQMapMessage::setFloat ( const std::string
            & name, float value ) throw ( cms::MessageNotWriteableException,
              cms::CMSEException ) [virtual]
```

Sets a Float value with the specified name into the Map.

#### Parameters

<i>name</i>	The name of the Float
<i>value</i>	The Float value to set in the Map

#### Exceptions

<i>CMSEException</i>	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i>	- if the <b>Message</b> (p. 2596) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2561).

```
6.37.2.30  virtual void activemq::commands::ActiveMQMapMessage::setInt ( const std::string
            & name, int value ) throw ( cms::MessageNotWriteableException,
              cms::CMSEException ) [virtual]
```

Sets a Int value with the specified name into the Map.

**Parameters**

<i>name</i>	The name of the Int
<i>value</i>	The Int value to set in the Map

**Exceptions**

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i>	- if the <b>Message</b> (p. 2596) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2561).

**6.37.2.31** `virtual void activemq::commands::ActiveMQMapMessage::setLong ( const std::string & name, long long value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Sets a Long value with the specified name into the Map.

**Parameters**

<i>name</i>	The name of the Long
<i>value</i>	The Long value to set in the Map

**Exceptions**

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i>	- if the <b>Message</b> (p. 2596) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2562).

**6.37.2.32** `virtual void activemq::commands::ActiveMQMapMessage::setShort ( const std::string & name, short value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Sets a Short value with the specified name into the Map.

**Parameters**

<i>name</i>	The name of the Short
<i>value</i>	The Short value to set in the Map

**Exceptions**

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i>	- if the <b>Message</b> (p. 2596) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2562).

```
6.37.2.33 virtual void activemq::commands::ActiveMQMapMessage::setString
( const std::string & name, const std::string & value ) throw (
cms::MessageNotWriteableException, cms::CMSException )
[virtual]
```

Sets a String value with the specified name into the Map.

#### Parameters

<i>name</i>	The name of the String
<i>value</i>	The String value to set in the Map

#### Exceptions

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i>	- if the <b>Message</b> (p. 2596) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2562).

```
6.37.2.34 virtual std::string activemq::commands::ActiveMQMapMessage::toString ( ) const
[virtual]
```

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 2611).

### 6.37.3 Field Documentation

```
6.37.3.1 const unsigned char activemq::commands::ActiveMQMapMessage::ID_ -
ACTIVEMQMAPMESSAGE = 25 [static]
```

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQMapMessage.h**

## 6.38 activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 364).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMess
```

## Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.38.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 364).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.38.2 Constructor & Destructor Documentation

6.38.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller ( ) [inline]`

6.38.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller ( ) [inline, virtual]`

### 6.38.3 Member Function Documentation

6.38.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.38.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.38.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## 6.38 activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller

### Class Reference 367

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2786).

**6.38.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2786).

**6.38.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2787).

**6.38.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2788).

**6.38.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2788).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h`



6.39 ac-

ativemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller

Class Reference

6.39 ~~ativemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller~~<sup>369</sup>

## Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 369).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarsha
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller:

## Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.39.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 369).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.39.2 Constructor & Destructor Documentation

6.39.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller ( ) [inline]`

6.39.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller ( ) [inline, virtual]`

### 6.39.3 Member Function Documentation

6.39.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.39.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.39.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

### 6.39 activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller

#### Class Reference 371

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2799).

**6.39.3.4** virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2800).

**6.39.3.5** virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::tightMarshal1 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2800).

**6.39.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2801).

**6.39.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## 6.40 activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller

**Class Reference** 373

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2802).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarshaller.h

## 6.40 activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 373).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMapMessageMarsha
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller**:

### Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.40.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 373).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.40.2 Constructor & Destructor Documentation

6.40.2.1 **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller**  
( ) [inline]

6.40.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller**  
( ) [inline, virtual]

### 6.40.3 Member Function Documentation

6.40.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.40.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

## 6.40 ac-

**activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller**

**Class Reference**

**375**

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.40.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2795).

**6.40.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2795).

6.40.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2796).

6.40.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2797).



## 6.41 ac-

ativemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller

### Class Reference

377

6.40.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::tightUnmarshal  
( OpenWireFormat \* wireFormat, commands::DataStructure \*  
dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \*  
bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

wireFormat	- describes the wire format of the broker.
dataStructure	- Object to be un-marshaled.
dataIn	- BinaryReader that provides that data.
bs	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

IOException	if an error occurs during the unmarshal.
-------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v4::MessageMarshaller**  
(p. 2797).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMapMessageMarshaller.h

## 6.41 activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 377).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarsha
```

Inheritance diagram for **ativemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller**:

### Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.41.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 377).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.41.2 Constructor & Destructor Documentation

6.41.2.1 **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller**  
( ) [inline]

6.41.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller**  
( ) [inline, virtual]

### 6.41.3 Member Function Documentation

6.41.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

## 6.41 activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller

### Class Reference

---

#### Returns

379

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.41.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::getDataStructureType ( ) const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.41.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2781).

**6.41.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2782).

```
6.41.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2783).

```
6.41.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink

## 6.42 activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller

### Class Reference

381

<i>bs</i>	- BooleanStream stream used to pack bits from the wire.
-----------	---

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2783).

**6.41.3.7** virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarshaller.h

## 6.42 activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 381).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarsha
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller:

## Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.42.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 381).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

6.42.2.1 **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller**  
( ) [inline]

6.42.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller**  
( ) [inline, virtual]

### 6.42.3 Member Function Documentation

6.42.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.42.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.42.3.3 **virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2790).

**6.42.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2791).

**6.42.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



## 6.42 ac-

**activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller**

### Class Reference

385

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2791).

**6.42.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2792).

**6.42.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2793).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller.h

## 6.43 activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 386).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMapMess
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller`:

### Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 386).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

## 6.43.2 Constructor & Destructor Documentation

6.43.2.1 **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller**  
 ( ) [inline]

6.43.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller**  
 ( ) [inline, virtual]

## 6.43.3 Member Function Documentation

6.43.3.1 **virtual commands::DataStructure\* ac-  
 tivemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.43.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.43.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::looseMarshal**  
 ( **OpenWireFormat \* wireFormat**, **commands::DataStructure**  
 \* **dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2804).

```
6.43.3.4 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataInputStream * dataIn ) throw (
 decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2804).

```
6.43.3.5 virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
 dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2805).

**6.43.3.6** virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::tightMarshal2 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2806).

**6.43.3.7** virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2806).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQMapMessageMarshaller.h**

**6.44 activemq::commands::ActiveMQMessage Class Reference**

```
#include <src/main/activemq/commands/ActiveMQMessage.h>
```

Inheritance diagram for **activemq::commands::ActiveMQMessage**:

**Public Member Functions**

- **ActiveMQMessage** ()
- virtual **~ActiveMQMessage** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaller share.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual **ActiveMQMessage** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual **cms::Message** \* **clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*

### Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQMESSAGE** = 23

### 6.44.1 Constructor & Destructor Documentation

6.44.1.1 **activemq::commands::ActiveMQMessage::ActiveMQMessage ( )**

6.44.1.2 **virtual activemq::commands::ActiveMQMessage::~~ActiveMQMessage ( )**  
[inline, virtual]

### 6.44.2 Member Function Documentation

6.44.2.1 **virtual cms::Message\* activemq::commands::ActiveMQMessage::clone ( ) const**  
[inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

#### Returns

new copy of this message

Implements **cms::Message** (p. 2620).

6.44.2.2 **virtual ActiveMQMessage\* activemq::commands::ActiveMQMessage::cloneDataStructure ( ) const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 2601).

6.44.2.3 **virtual void activemq::commands::ActiveMQMessage::copyDataStructure ( const DataStructure \* src )** [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 2602).

**6.44.2.4** `virtual bool activemq::commands::ActiveMQMessage::equals ( const DataStructure  
* value ) const` [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::Message  
>` (p. 423).

**6.44.2.5** `virtual unsigned char activemq::commands::ActiveMQMessage::getDataStructureType ( ) const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1713) type copy.

Reimplemented from `activemq::commands::Message` (p. 2604).

**6.44.2.6** `virtual std::string activemq::commands::ActiveMQMessage::toString ( ) const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::Message` (p. 2611).

### 6.44.3 Field Documentation

**6.44.3.1** `const unsigned char activemq::commands::ActiveMQMessage::ID_ -  
ACTIVEMQMESSAGE = 23` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQMessage.h`



6.45

activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller

Class Reference

6.45 — activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller <sup>393</sup>

## Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 393).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller:

## Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual **~ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.45.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 393).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.45.2 Constructor & Destructor Documentation

6.45.2.1 **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller**  
 ( ) [inline]

6.45.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller**  
 ( ) [inline, virtual]

### 6.45.3 Member Function Documentation

6.45.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.45.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.45.3.3 **virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::looseMarshal**  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
 \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw (  
**decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

## 6.45

### activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller

#### Class Reference

395

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2786).

**6.45.3.4** virtual void **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2786).

**6.45.3.5** virtual int **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::tightMarshal1** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2787).

**6.45.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2788).

**6.45.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## 6.46

**activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller**

**Class Reference**

397

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2788).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h`

## 6.46 **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 397).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller**:

### Public Member Functions

- **ActiveMQMessageMarshaller ()**
- **virtual ~ActiveMQMessageMarshaller ()**
- **virtual commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- **virtual unsigned char getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaller.*
- **virtual void tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*
- **virtual int tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- **virtual void tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.46.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 397).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.46.2 Constructor & Destructor Documentation

6.46.2.1 **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller**  
( ) [inline]

6.46.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller**  
( ) [inline, virtual]

### 6.46.3 Member Function Documentation

6.46.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.46.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

## 6.46

**activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller**

**Class Reference**

**399**

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.46.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2799).

**6.46.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2800).

6.46.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2800).

6.46.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2801).



## 6.47

**activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller**

### Class Reference

401

6.46.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2802).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h`

## 6.47 **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 401).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMessageMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller**:

### Public Member Functions

- **ActiveMQMessageMarshaller ()**
- **virtual ~ActiveMQMessageMarshaller ()**
- **virtual commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- **virtual unsigned char getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.47.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 401).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.47.2 Constructor & Destructor Documentation

6.47.2.1 **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller**  
( ) [inline]

6.47.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller**  
( ) [inline, virtual]

### 6.47.3 Member Function Documentation

6.47.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

6.47

**activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller**

**Class Reference**

**403**

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.47.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.47.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2795).

**6.47.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2795).

```
6.47.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2796).

```
6.47.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink

## 6.48

### activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller Class Reference 405

<i>bs</i>	- BooleanStream stream used to pack bits from the wire.
-----------	---

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2797).

**6.47.3.7** virtual void **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2797).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMessageMarshaller.h

## 6.48 activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 405).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMessageMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller**:

## Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual **~ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**)  
 throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.48.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 405).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

**6.48.2 Constructor & Destructor Documentation**

**6.48.2.1** `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller ( ) [inline]`

**6.48.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller ( ) [inline, virtual]`

**6.48.3 Member Function Documentation**

**6.48.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

**6.48.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.48.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2781).

**6.48.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2782).

**6.48.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



## 6.48

**activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller**

### Class Reference

409

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2783).

**6.48.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2783).

**6.48.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2784).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMessageMarshaller.h`

## 6.49 activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 410).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller`:

### Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual **~ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

6.49

activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller

Class Reference

411

### 6.49.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 410).  
NOTE!: This file is auto generated - do not modify! if you need to make a change,  
please see the Java Classes in the activemq-openwire-generator module

### 6.49.2 Constructor & Destructor Documentation

6.49.2.1 **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller**  
( ) [inline]

6.49.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller**  
( ) [inline, virtual]

### 6.49.3 Member Function Documentation

6.49.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.49.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.49.3.3 **virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2790).

**6.49.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2791).

**6.49.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## 6.49

### activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller

#### Class Reference

413

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2791).

**6.49.3.6** virtual void **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::tightMarshal2** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2792).

**6.49.3.7** virtual void **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2793).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h`

## 6.50 `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 414).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller`:

**Public Member Functions**

- **ActiveMQMessageMarshaller** ()
- virtual **~ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

## 6.50

**activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller**

### Class Reference

415

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.50.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 414).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.50.2 Constructor & Destructor Documentation

6.50.2.1 **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller**  
( ) [inline]

6.50.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller**  
( ) [inline, virtual]

### 6.50.3 Member Function Documentation

6.50.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.50.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.50.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2804).

**6.50.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2804).



6.50

activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller

Class Reference

417

6.50.3.5 virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::tightMarshal1  
( OpenWireFormat \* wireFormat, commands::DataStructure \*  
dataStructure, utils::BooleanStream \* bs ) throw ( decaf::io::IOException )  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

wireFormat	- describes the wire format of the broker
dataStructure	- Object to be marshaled
bs	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

IOException	if an error occurs during the marshal.
-------------	--

Reimplemented from activemq::wireformat::openwire::marshal::v6::MessageMarshaller  
(p. 2805).

6.50.3.6 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::tightMarshal2  
( OpenWireFormat \* wireFormat, commands::DataStructure  
\* dataStructure, decaf::io::DataOutputStream \* dataOut,  
utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

wireFormat	- describes the wire format of the broker
dataStructure	- Object to be marshaled
dataOut	- BinaryReader that provides that data sink
bs	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

IOException	if an error occurs during the marshal.
-------------	--

Reimplemented from activemq::wireformat::openwire::marshal::v6::MessageMarshaller  
(p. 2806).

```
6.50.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2806).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMessageMarshaller.h`

## 6.51 `activemq::commands::ActiveMQMessageTemplate< T >` Class Template Reference

```
#include <src/main/activemq/commands/ActiveMQMessageTemplate.h>
```

Inheritance diagram for `activemq::commands::ActiveMQMessageTemplate< T >`:

#### Public Member Functions

- `ActiveMQMessageTemplate ()`
- `virtual ~ActiveMQMessageTemplate ()`
- `virtual void acknowledge () const throw ( cms::IllegalStateException, cms::CMSException )`  
*Acknowledges all consumed messages of the session of this consumed message.*
- `virtual void onSend ()`  
*Resets the **Message** (p. 2596) to a Read-Only state.*
- `virtual bool equals (const DataStructure *value) const`

*Compares the **DataSet** (p. 1713) passed in to this one, and returns if they are equivalent.*

- virtual void **clearBody** () throw ( cms::CMSEException )  
*Clears out the body of the message.*
- virtual void **clearProperties** () throw ( cms::CMSEException )  
*Clears the message properties.*
- virtual std::vector< std::string > **getPropertyNames** () const throw ( cms::CMSEException )  
*Retrieves the property names.*
- virtual bool **propertyExists** (const std::string &name) const throw ( cms::CMSEException )  
*Indicates whether or not a given property exists.*
- virtual bool **getBooleanProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Gets a boolean property.*
- virtual unsigned char **getByteProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Gets a byte property.*
- virtual double **getDoubleProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Gets a double property.*
- virtual float **getFloatProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Gets a float property.*
- virtual int **getIntProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Gets a int property.*
- virtual long long **getLongProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Gets a long property.*
- virtual short **getShortProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Gets a short property.*
- virtual std::string **getStringProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )

*Gets a string property.*

- virtual void **setBooleanProperty** (const std::string &name, bool value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a boolean property.*

- virtual void **setByteProperty** (const std::string &name, unsigned char value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a byte property.*

- virtual void **setDoubleProperty** (const std::string &name, double value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a double property.*

- virtual void **setFloatProperty** (const std::string &name, float value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a float property.*

- virtual void **setIntProperty** (const std::string &name, int value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a int property.*

- virtual void **setLongProperty** (const std::string &name, long long value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a long property.*

- virtual void **setShortProperty** (const std::string &name, short value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a short property.*

- virtual void **setStringProperty** (const std::string &name, const std::string &value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a string property.*

- virtual std::string **getCMSCorrelationID** () const throw ( cms::CMSEException )

*Get the Correlation Id for this message.*

- virtual void **setCMSCorrelationID** (const std::string &correlationId) throw ( cms::CMSEException )

*Sets the Correlation Id used by this message.*

- virtual int **getCMSDeliveryMode** () const throw ( cms::CMSEException )

*Gets the DeliveryMode for this message.*

- virtual void **setCMSDeliveryMode** (int mode) throw ( cms::CMSEException )

*Sets the DeliveryMode for this message.*

- virtual const **cms::Destination** \* **getCMSDestination** () const throw ( cms::CMSException )  
*Gets the Destination for this Message (p. 2596), returns a.*
- virtual void **setCMSDestination** (const **cms::Destination** \***destination**) throw ( cms::CMSException )  
*Sets the Destination for this message.*
- virtual long long **getCMSExpiration** () const throw ( cms::CMSException )  
*Gets the Expiration Time for this Message (p. 2596).*
- virtual void **setCMSExpiration** (long long expireTime) throw ( cms::CMSException )  
*Sets the Expiration Time for this message.*
- virtual std::string **getCMSMessageID** () const throw ( cms::CMSException )  
*Gets the CMS Message (p. 2596) Id for this Message (p. 2596).*
- virtual void **setCMSMessageID** (const std::string &id AMQCPP\_UNUSED) throw ( cms::CMSException )  
*Sets the CMS Message (p. 2596) Id for this message.*
- virtual int **getCMSPriority** () const throw ( cms::CMSException )  
*Gets the Priority Value for this Message (p. 2596).*
- virtual void **setCMSPriority** (int **priority**) throw ( cms::CMSException )  
*Sets the Priority Value for this message.*
- virtual bool **getCMSRedelivered** () const throw ( cms::CMSException )  
*Gets the Redelivered Flag for this Message (p. 2596).*
- virtual void **setCMSRedelivered** (bool redelivered AMQCPP\_UNUSED) throw ( cms::CMSException )  
*Sets the Redelivered Flag for this message.*
- virtual const **cms::Destination** \* **getCMSReplyTo** () const throw ( cms::CMSException )  
*Gets the CMS Reply To Address for this Message (p. 2596).*
- virtual void **setCMSReplyTo** (const **cms::Destination** \***destination**) throw ( cms::CMSException )  
*Sets the CMS Reply To Address for this message.*
- virtual long long **getCMSTimestamp** () const throw ( cms::CMSException )  
*Gets the Time Stamp for this Message (p. 2596).*

- virtual void **setCMSTimestamp** (long long timeStamp) throw ( cms::CMSEException )  
*Sets the Time Stamp for this message.*
- virtual std::string **getCMSType** () const throw ( cms::CMSEException )  
*Gets the CMS Message (p. 2596) Type for this Message (p. 2596).*
- virtual void **setCMSType** (const std::string &type) throw ( cms::CMSEException )  
*Sets the CMS Message (p. 2596) Type for this message.*

### Protected Member Functions

- void **failIfWriteOnlyBody** () const
- void **failIfReadOnlyBody** () const
- void **failIfReadOnlyProperties** () const

```
template<typename T> class activemq::commands::ActiveMQMessageTemplate< T >
```

### 6.51.1 Constructor & Destructor Documentation

6.51.1.1 `template<typename T> activemq::commands::ActiveMQMessageTemplate< T >::ActiveMQMessageTemplate ( ) [inline]`

6.51.1.2 `template<typename T> virtual activemq::commands::ActiveMQMessageTemplate< T >::~~ActiveMQMessageTemplate ( ) [inline, virtual]`

### 6.51.2 Member Function Documentation

6.51.2.1 `template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::acknowledge ( ) const throw ( cms::IllegalStateException, cms::CMSEException ) [inline, virtual]`

Acknowledges all consumed messages of the session of this consumed message.

6.51.2.2 `template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::clearBody ( ) throw ( cms::CMSEException ) [inline, virtual]`

Clears out the body of the message.

This does not clear the headers or properties.

Reimplemented in `activemq::commands::ActiveMQBytesMessage` (p. 219), `activemq::commands::ActiveMQTextMessage` (p. 354), `activemq::commands::ActiveMQStreamMessage` (p. 539), and `activemq::commands::ActiveMQTopicMessage` (p. 668).

**6.51.2.3** `template<typename T> virtual void  
 activemq::commands::ActiveMQMessageTemplate< T  
 >::clearProperties ( ) throw ( cms::CMSException ) [inline,  
 virtual]`

Clears the message properties.

Does not clear the body or header values.

**6.51.2.4** `template<typename T> virtual bool  
 activemq::commands::ActiveMQMessageTemplate< T  
 >::equals ( const DataStructure * value ) const [inline, virtual]`

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::Message** (p. 2602).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 187), **activemq::commands::ActiveMQBytesMessage** (p. 220), **activemq::commands::ActiveMQMapMessage** (p. 355), **activemq::commands::ActiveMQMessage** (p. 392), **activemq::commands::ActiveMQObjectMessage** (p. 439), **activemq::commands::ActiveMQStreamMessage** (p. 540), and **activemq::commands::ActiveMQTextMessage** (p. 669).

**6.51.2.5** `template<typename T> void activemq::commands::ActiveMQMessageTemplate<  
 T >::failIfReadOnlyBody ( ) const [inline, protected]`

**6.51.2.6** `template<typename T> void activemq::commands::ActiveMQMessageTemplate<  
 T >::failIfReadOnlyProperties ( ) const [inline, protected]`

**6.51.2.7** `template<typename T> void activemq::commands::ActiveMQMessageTemplate<  
 T >::failIfWriteOnlyBody ( ) const [inline, protected]`

**6.51.2.8** `template<typename T> virtual bool  
 activemq::commands::ActiveMQMessageTemplate< T  
 >::getBooleanProperty ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSException ) [inline,  
 virtual]`

Gets a boolean property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

**Returns**

The value for the named property.

**Exceptions**

<i>CMSEException</i>	if the property does not exist.
<i>MessageFormatException</i>	- if this type conversion is invalid.

```
6.51.2.9  template<typename T> virtual unsigned char
          activemq::commands::ActiveMQMessageTemplate< T >::getBytesProperty
          ( const std::string & name ) const throw ( cms::MessageFormatException,
          cms::CMSEException ) [inline, virtual]
```

Gets a byte property.

**Parameters**

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

**Returns**

The value for the named property.

**Exceptions**

<i>CMSEException</i>	if the property does not exist.
<i>MessageFormatException</i>	- if this type conversion is invalid.

```
6.51.2.10 template<typename T> virtual std::string
          activemq::commands::ActiveMQMessageTemplate< T
          >::getCMSCorrelationID ( ) const throw ( cms::CMSEException ) [inline,
          virtual]
```

Get the Correlation Id for this message.

**Returns**

string representation of the correlation Id

**Exceptions**

<i>CMSEException</i>	
----------------------	--



**6.51.2.11** `template<typename T> virtual int  
activemq::commands::ActiveMQMessageTemplate< T  
>::getCMSDeliveryMode ( ) const throw ( cms::CMSException ) [inline,  
virtual]`

Gets the DeliveryMode for this message.

#### Returns

DeliveryMode enumerated value.

#### Exceptions

<i>CMSException</i>	
---------------------	--

**6.51.2.12** `template<typename T> virtual const cms::Destination*  
activemq::commands::ActiveMQMessageTemplate< T  
>::getCMSDestination ( ) const throw ( cms::CMSException ) [inline,  
virtual]`

Gets the Destination for this **Message** (p. 2596), returns a.

#### Returns

Destination object

#### Exceptions

<i>CMSException</i>	
---------------------	--

**6.51.2.13** `template<typename T> virtual long long  
activemq::commands::ActiveMQMessageTemplate< T  
>::getCMSExpiration ( ) const throw ( cms::CMSException ) [inline,  
virtual]`

Gets the Expiration Time for this **Message** (p. 2596).

#### Returns

time value

#### Exceptions

<i>CMSException</i>	
---------------------	--

**6.51.2.14** `template<typename T> virtual std::string  
activemq::commands::ActiveMQMessageTemplate< T  
>::getCMSMessageID ( ) const throw ( cms::CMSEException ) [inline,  
virtual]`

Gets the CMS **Message** (p. 2596) Id for this **Message** (p. 2596).

#### Returns

time value

#### Exceptions

<i>CMSEException</i>
----------------------

**6.51.2.15** `template<typename T> virtual int  
activemq::commands::ActiveMQMessageTemplate< T  
>::getCMSPriority ( ) const throw ( cms::CMSEException ) [inline,  
virtual]`

Gets the Priority Value for this **Message** (p. 2596).

#### Returns

priority value

#### Exceptions

<i>CMSEException</i>
----------------------

**6.51.2.16** `template<typename T> virtual bool  
activemq::commands::ActiveMQMessageTemplate< T  
>::getCMSRedelivered ( ) const throw ( cms::CMSEException ) [inline,  
virtual]`

Gets the Redelivered Flag for this **Message** (p. 2596).

#### Returns

redelivered value

#### Exceptions

<i>CMSEException</i>
----------------------

6.51.2.17 `template<typename T> virtual const cms::Destination*  
activemq::commands::ActiveMQMessageTemplate< T >::getCMSReplyTo  
( ) const throw ( cms::CMSException ) [inline, virtual]`

Gets the CMS Reply To Address for this **Message** (p. 2596).

#### Returns

Reply To Value

#### Exceptions

<i>CMSException</i>	
---------------------	--

6.51.2.18 `template<typename T> virtual long long  
activemq::commands::ActiveMQMessageTemplate< T  
>::getCMSTimestamp ( ) const throw ( cms::CMSException ) [inline,  
virtual]`

Gets the Time Stamp for this **Message** (p. 2596).

#### Returns

time stamp value

#### Exceptions

<i>CMSException</i>	
---------------------	--

6.51.2.19 `template<typename T> virtual std::string  
activemq::commands::ActiveMQMessageTemplate< T  
>::getCMSType ( ) const throw ( cms::CMSException ) [inline,  
virtual]`

Gets the CMS **Message** (p. 2596) Type for this **Message** (p. 2596).

#### Returns

type value

#### Exceptions

<i>CMSException</i>	
---------------------	--

```

6.51.2.20  template<typename T> virtual double
            activemq::commands::ActiveMQMessageTemplate< T
            >::getDoubleProperty ( const std::string & name ) const throw (
            cms::MessageFormatException, cms::CMSEException ) [inline,
            virtual]

```

Gets a double property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<i>CMSEException</i>	if the property does not exist.
<i>MessageFormatException</i>	- if this type conversion is invalid.

```

6.51.2.21  template<typename T> virtual float
            activemq::commands::ActiveMQMessageTemplate< T
            >::getFloatProperty ( const std::string & name ) const throw (
            cms::MessageFormatException, cms::CMSEException ) [inline,
            virtual]

```

Gets a float property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<i>CMSEException</i>	if the property does not exist.
<i>MessageFormatException</i>	- if this type conversion is invalid.

6.51.2.22 `template<typename T> virtual int  
activemq::commands::ActiveMQMessageTemplate<  
T >::getIntProperty ( const std::string & name ) const throw (  
cms::MessageFormatException, cms::CMSEException ) [inline,  
virtual]`

Gets a int property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<i>CMSEException</i>	if the property does not exist.
<i>MessageFormatEx- ception</i>	- if this type conversion is invalid.

6.51.2.23 `template<typename T> virtual long long  
activemq::commands::ActiveMQMessageTemplate< T  
>::getLongProperty ( const std::string & name ) const throw (  
cms::MessageFormatException, cms::CMSEException ) [inline,  
virtual]`

Gets a long property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<i>CMSEException</i>	if the property does not exist.
<i>MessageFormatEx- ception</i>	- if this type conversion is invalid.

```
6.51.2.24 template<typename T> virtual std::vector<std::string>
activemq::commands::ActiveMQMessageTemplate< T
>::getPropertyNames ( ) const throw ( cms::CMSException ) [inline,
virtual]
```

Retrieves the property names.

#### Returns

The complete set of property names currently in this message.

```
6.51.2.25 template<typename T> virtual short
activemq::commands::ActiveMQMessageTemplate< T
>::getShortProperty ( const std::string & name ) const throw (
cms::MessageFormatException, cms::CMSException ) [inline,
virtual]
```

Gets a short property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<i>CMSException</i>	if the property does not exist.
<i>MessageFormatException</i>	- if this type conversion is invalid.

```
6.51.2.26 template<typename T> virtual std::string
activemq::commands::ActiveMQMessageTemplate< T
>::getStringProperty ( const std::string & name ) const throw (
cms::MessageFormatException, cms::CMSException ) [inline,
virtual]
```

Gets a string property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

## Exceptions

<i>CMSEException</i>	if the property does not exist.
<i>MessageFormatException</i>	- if this type conversion is invalid.

**6.51.2.27** `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::onSend ( ) [inline, virtual]`

Resets the **Message** (p. 2596) to a Read-Only state.

Reimplemented from **activemq::commands::Message** (p. 2608).

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 221), and **activemq::commands::ActiveMQStreamMessage** (p. 540).

**6.51.2.28** `template<typename T> virtual bool  
activemq::commands::ActiveMQMessageTemplate< T  
>::propertyExists ( const std::string & name ) const throw ( cms::CMSEException  
) [inline, virtual]`

Indicates whether or not a given property exists.

## Parameters

<i>name</i>	The name of the property to look up.
-------------	--------------------------------------

## Returns

True if the property exists in this message.

**6.51.2.29** `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::setBooleanProperty ( const std::string & name, bool value ) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
[inline, virtual]`

Sets a boolean property.

## Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

## Exceptions

<i>CMSEException</i>	- if the name is an empty string.
<i>MessageNotWriteableException</i>	- if properties are read-only

```

6.51.2.30 template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setByteProperty ( const std::string & name, unsigned char value ) throw (
cms::MessageNotWriteableException, cms::CMSException )
[inline, virtual]

```

Sets a byte property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<i>CMSException</i>	- if the name is an empty string.
<i>MessageNotWriteableException</i>	- if properties are read-only

```

6.51.2.31 template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setCMSCorrelationID ( const std::string & correlationId ) throw (
cms::CMSException ) [inline, virtual]

```

Sets the Correlation Id used by this message.

#### Parameters

<i>correlationId</i>	- String representing the correlation id.
----------------------	---

#### Exceptions

<i>CMSException</i>	
---------------------	--

```

6.51.2.32 template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setCMSDeliveryMode ( int mode ) throw ( cms::CMSException )
[inline, virtual]

```

Sets the DeliveryMode for this message.

#### Parameters

<i>mode</i>	- DeliveryMode enumerated value.
-------------	----------------------------------

#### Exceptions

<i>CMSException</i>	
---------------------	--



6.51.2.33 `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::setCMSDestination ( const cms::Destination * destination ) throw ( cms::CMSException ) [inline, virtual]`

Sets the Destination for this message.

#### Parameters

<i>destination</i>	- Destination Object
--------------------	----------------------

#### Exceptions

<i>CMSException</i>	
---------------------	--

6.51.2.34 `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::setCMSExpiration ( long long expireTime ) throw ( cms::CMSException )  
[inline, virtual]`

Sets the Expiration Time for this message.

#### Parameters

<i>expireTime</i>	- time value
-------------------	--------------

#### Exceptions

<i>CMSException</i>	
---------------------	--

6.51.2.35 `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::setCMSMessageID ( const std::string &id AMQCPP_UNUSED ) throw ( cms::CMSException ) [inline, virtual]`

Sets the CMS **Message** (p. 2596) Id for this message.

#### Parameters

<i>id</i>	- time value
-----------	--------------

#### Exceptions

<i>CMSException</i>	
---------------------	--

```
6.51.2.36  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setCMSPriority ( int priority ) throw ( cms::CMSEException )  [inline,
           virtual]
```

Sets the Priority Value for this message.

#### Parameters

<i>priority</i>	- priority value for this message
-----------------	-----------------------------------

#### Exceptions

<i>CMSEException</i>
----------------------

```
6.51.2.37  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setCMSRedelivered ( bool redelivered AMQCPP_UNUSED ) throw (
           cms::CMSEException )  [inline, virtual]
```

Sets the Redelivered Flag for this message.

#### Parameters

<i>redelivered</i>	- boolean redelivered value
--------------------	-----------------------------

#### Exceptions

<i>CMSEException</i>
----------------------

```
6.51.2.38  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setCMSReplyTo ( const cms::Destination * destination ) throw (
           cms::CMSEException )  [inline, virtual]
```

Sets the CMS Reply To Address for this message.

#### Parameters

<i>destination</i>	Pointer to the CMS Destination that is the Reply-To value.
--------------------	--

#### Exceptions

<i>CMSEException</i>
----------------------

6.51.2.39 `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::setCMSTimestamp ( long long timeStamp ) throw ( cms::CMSEException )  
[inline, virtual]`

Sets the Time Stamp for this message.

#### Parameters

<i>timeStamp</i>	- integer time stamp value
------------------	----------------------------

#### Exceptions

<i>CMSEException</i>	
----------------------	--

6.51.2.40 `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::setCMSType ( const std::string & type ) throw ( cms::CMSEException )  
[inline, virtual]`

Sets the CMS **Message** (p. 2596) Type for this message.

#### Parameters

<i>type</i>	- message type value string
-------------	-----------------------------

#### Exceptions

<i>CMSEException</i>	
----------------------	--

6.51.2.41 `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::setDoubleProperty ( const std::string & name, double value ) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
[inline, virtual]`

Sets a double property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<i>CMSEException</i>	- if the name is an empty string.
<i>MessageNotWriteableException</i>	- if properties are read-only

```

6.51.2.42  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setFloatProperty ( const std::string & name, float value ) throw (
            cms::MessageNotWriteableException, cms::CMSException )
            [inline, virtual]

```

Sets a float property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<i>CMSException</i>	- if the name is an empty string.
<i>MessageNotWriteableException</i>	- if properties are read-only

```

6.51.2.43  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setIntProperty ( const std::string & name, int value ) throw (
            cms::MessageNotWriteableException, cms::CMSException )
            [inline, virtual]

```

Sets a int property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<i>CMSException</i>	- if the name is an empty string.
<i>MessageNotWriteableException</i>	- if properties are read-only

```

6.51.2.44  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setLongProperty ( const std::string & name, long long value ) throw (
            cms::MessageNotWriteableException, cms::CMSException )
            [inline, virtual]

```

Sets a long property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

<i>value</i>	The value for the named property.
--------------	-----------------------------------

#### Exceptions

<i>CMSEException</i>	- if the name is an empty string.
<i>MessageNotWriteableException</i>	- if properties are read-only

**6.51.2.45** `template<typename T> virtual void  
 activemq::commands::ActiveMQMessageTemplate< T  
 >::setShortProperty ( const std::string & name, short value ) throw (   
 cms::MessageNotWriteableException, cms::CMSEException )  
 [inline, virtual]`

Sets a short property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<i>CMSEException</i>	- if the name is an empty string.
<i>MessageNotWriteableException</i>	- if properties are read-only

**6.51.2.46** `template<typename T> virtual void  
 activemq::commands::ActiveMQMessageTemplate< T  
 >::setStringProperty ( const std::string & name, const std::string & value )  
 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
 [inline, virtual]`

Sets a string property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<i>CMSEException</i>	- if the name is an empty string.
<i>MessageNotWriteableException</i>	- if properties are read-only

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQMessageTemplate.h**

## 6.52 activemq::commands::ActiveMQObjectMessage Class Reference

```
#include <src/main/activemq/commands/ActiveMQObjectMessage.h>
```

Inheritance diagram for activemq::commands::ActiveMQObjectMessage:

### Public Member Functions

- **ActiveMQObjectMessage** ()
- virtual **~ActiveMQObjectMessage** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ActiveMQObjectMessage \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual **cms::Message \* clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*

### Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQOBJECTMESSAGE** = 26

### 6.52.1 Constructor & Destructor Documentation

6.52.1.1 `activemq::commands::ActiveMQObjectMessage::ActiveMQObjectMessage ( )`

6.52.1.2 `virtual activemq::commands::ActiveMQObjectMessage::~~ActiveMQObjectMessage ( )`  
[inline, virtual]

### 6.52.2 Member Function Documentation

6.52.2.1 `virtual cms::Message* activemq::commands::ActiveMQObjectMessage::clone ( )`  
`const` [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

#### Returns

new copy of this message

Implements `cms::Message` (p. 2620).

6.52.2.2 `virtual ActiveMQObjectMessage* activemq::commands::ActiveMQObjectMessage::cloneDataStructure ( ) const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2601).

6.52.2.3 `virtual void activemq::commands::ActiveMQObjectMessage::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2602).

6.52.2.4 `virtual bool activemq::commands::ActiveMQObjectMessage::equals ( const DataStructure * value ) const` [virtual]

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if **DataSet**'s are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 423).

**6.52.2.5** `virtual unsigned char activemq::commands::ActiveMQObjectMessage::getDataSetType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaller share.

### Returns

new **DataSet** (p. 1713) type copy.

Reimplemented from **activemq::commands::Message** (p. 2604).

**6.52.2.6** `virtual std::string activemq::commands::ActiveMQObjectMessage::toString ( ) const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1713) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 2611).

## 6.52.3 Field Documentation

**6.52.3.1** `const unsigned char activemq::commands::ActiveMQObjectMessage::ID_ACTIVEMQOBJECTMESSAGE = 26 [static]`

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQObjectMessage.h**

## 6.53 **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 440).



## 6.53 ac-

**activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller**

**Class Reference**

**441**

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMar
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller`:

### Public Member Functions

- **ActiveMQObjectMessageMarshaller ()**
- virtual **~ActiveMQObjectMessageMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.53.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 440). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

### 6.53.2 Constructor & Destructor Documentation

6.53.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller ( ) [inline]`

6.53.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller ( ) [inline, virtual]`

### 6.53.3 Member Function Documentation

6.53.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.53.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.53.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## 6.53 activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller

### Class Reference

### Exceptions

443

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2786).

**6.53.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2786).

**6.53.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2787).

**6.53.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2788).

**6.53.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2788).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMarshaller.h`

6.54 ac-

ativemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller

Class Reference

6.54 activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller

445

## Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller**  
(p. 445).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMar
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller:

## Public Member Functions

- **ActiveMQObjectMessageMarshaller ()**
- virtual **~ActiveMQObjectMessageMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)**  
throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn)** throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.54.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 445). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.54.2 Constructor & Destructor Documentation

6.54.2.1 **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller**  
( ) [inline]

6.54.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller**  
( ) [inline, virtual]

### 6.54.3 Member Function Documentation

6.54.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.54.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.54.3.3 **virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::looseMarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

## 6.54 activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller

### Class Reference 447

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2799).

**6.54.3.4** virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2800).

**6.54.3.5** virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::tightMarshal1 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2800).

**6.54.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2801).

**6.54.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.



## 6.55 activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller

### Class Reference

### Exceptions

449

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2802).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMarshaller.h

## 6.55 activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 449).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQObjectMessageMar
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller**:

### Public Member Functions

- **ActiveMQObjectMessageMarshaller ()**
- virtual **~ActiveMQObjectMessageMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.55.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 449). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.55.2 Constructor & Destructor Documentation

- 6.55.2.1 **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller**  
 ( ) [inline]
- 6.55.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller**  
 ( ) [inline, virtual]

### 6.55.3 Member Function Documentation

- 6.55.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

- 6.55.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

## 6.55 ac-

**activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller**

**Class Reference**

**451**

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.55.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2795).

**6.55.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2795).

```
6.55.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2796).

```
6.55.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2797).

## 6.56 ac-

**activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller**

### Class Reference

453

```
6.55.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::tightUnmarshal  
( OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *  
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2797).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQObjectMessageMarshaller.h

## 6.56 activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 453).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMar
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller**:

### Public Member Functions

- **ActiveMQObjectMessageMarshaller ()**
- virtual **~ActiveMQObjectMessageMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.56.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 453). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.56.2 Constructor & Destructor Documentation

- 6.56.2.1 **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller**  
( ) [inline]
- 6.56.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller**  
( ) [inline, virtual]

### 6.56.3 Member Function Documentation

- 6.56.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.56.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::getDataStructureType ( ) const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.56.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2781).

**6.56.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2782).

```
6.56.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2783).

```
6.56.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink



## 6.57 activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller Class Reference 457

<i>bs</i>	- BooleanStream stream used to pack bits from the wire.
-----------	---

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2783).

**6.56.3.7** virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMarshaller.h

## 6.57 activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 457).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMar
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller:

## Public Member Functions

- **ActiveMQObjectMessageMarshaller ()**
- virtual **~ActiveMQObjectMessageMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.57.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 457). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.57.2 Constructor & Destructor Documentation

6.57.2.1 **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller**  
( ) [inline]

6.57.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller**  
( ) [inline, virtual]

## 6.57.3 Member Function Documentation

6.57.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.57.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.57.3.3 **virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2790).

**6.57.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2791).

**6.57.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.57 ac-

**activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller**

### Class Reference

461

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2791).

**6.57.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2792).

**6.57.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2793).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h

## 6.58 activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 462).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQObjectM
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller`:

### Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual **~ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 462). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.58.2 Constructor & Destructor Documentation

6.58.2.1 **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller**  
 ( ) [inline]

6.58.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller**  
 ( ) [inline, virtual]

## 6.58.3 Member Function Documentation

6.58.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.58.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.58.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::looseMarshal**  
 ( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2804).

**6.58.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2804).

**6.58.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2805).

**6.58.3.6** virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::tightMarshal2 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2806).

**6.58.3.7** virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2806).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQObjectMessageMarshaller.h

**6.59 activemq::core::ActiveMQProducer Class Reference**

```
#include <src/main/activemq/core/ActiveMQProducer.h>
```

Inheritance diagram for **activemq::core::ActiveMQProducer**:

**Public Member Functions**

- **ActiveMQProducer** (**ActiveMQSession** \*session, const **Pointer**< **commands::ProducerId** > &producerId, const **Pointer**< **commands::ActiveMQDestination** > &destination, long long sendTimeout)  
*Constructor, creates an instance of an ActiveMQProducer (p. 466).*
- virtual ~**ActiveMQProducer** ()
- virtual void **close** () throw ( cms::CMSException )  
*Closes the Consumer.*
- virtual void **send** (cms::Message \*message) throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )  
*Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **send** (cms::Message \*message, int deliveryMode, int priority, long long timeToLive) throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )  
*Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **send** (const cms::Destination \*destination, cms::Message \*message) throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )  
*Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.*

- virtual void **send** (const **cms::Destination** \*destination, **cms::Message** \*message, int deliveryMode, int priority, long long timeToLive) throw ( cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )  
*Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **setDeliveryMode** (int mode) throw ( cms::CMSEException )  
*Sets the delivery mode for this Producer.*
- virtual int **getDeliveryMode** () const throw ( cms::CMSEException )  
*Gets the delivery mode for this Producer.*
- virtual void **setDisableMessageID** (bool value) throw ( cms::CMSEException )  
*Sets if Message Ids are disabled for this Producer.*
- virtual bool **getDisableMessageID** () const throw ( cms::CMSEException )  
*Gets if Message Ids are disabled for this Producer.*
- virtual void **setDisableMessageTimeStamp** (bool value) throw ( cms::CMSEException )  
*Sets if Message Time Stamps are disabled for this Producer.*
- virtual bool **getDisableMessageTimeStamp** () const throw ( cms::CMSEException )  
*Gets if Message Time Stamps are disabled for this Producer.*
- virtual void **setPriority** (int priority) throw ( cms::CMSEException )  
*Sets the Priority that this Producers sends messages at.*
- virtual int **getPriority** () const throw ( cms::CMSEException )  
*Gets the Priority level that this producer sends messages at.*
- virtual void **setTimeToLive** (long long time) throw ( cms::CMSEException )  
*Sets the Time to Live that this Producers sends messages with.*
- virtual long long **getTimeToLive** () const throw ( cms::CMSEException )  
*Gets the Time to Live that this producer sends messages with.*
- virtual void **setSendTimeout** (long long time) throw ( cms::CMSEException )  
*Sets the Send Timeout that this Producers sends messages with.*
- virtual long long **getSendTimeout** () const throw ( cms::CMSEException )  
*Gets the Send Timeout that this producer sends messages with.*
- bool **isClosed** () const
- const **Pointer**< **commands::ProducerInfo** > & **getProducerInfo** () const

*Retries this object ProducerInfo pointer.*

- const **Pointer**< **commands::ProducerId** > & **getProducerId** () const  
*Retries this object ProducerId or NULL if closed.*
- virtual void **onProducerAck** (const **commands::ProducerAck** &ack)  
*Handles the work of Processing a ProducerAck Command from the Broker.*

## 6.59.1 Constructor & Destructor Documentation

**6.59.1.1** **activemq::core::ActiveMQProducer::ActiveMQProducer** ( **ActiveMQSession** \*  
**session**, const **Pointer**< **commands::ProducerId** > & **producerId**, const  
**Pointer**< **commands::ActiveMQDestination** > & **destination**, long long  
**sendTimeout** )

Constructor, creates an instance of an **ActiveMQProducer** (p. 466).

### Parameters

<i>session</i>	The Session which is the parent of this Producer.
<i>producerId</i>	Pointer to a ProducerId object which identifies this producer.
<i>destination</i>	The assigned Destination this Producer sends to, or null if not set. The Producer does not own the Pointer passed.
<i>sendTimeout</i>	The configured send timeout for this Producer.

**6.59.1.2** **virtual activemq::core::ActiveMQProducer::~~ActiveMQProducer** ( ) [virtual]

## 6.59.2 Member Function Documentation

**6.59.2.1** **virtual void activemq::core::ActiveMQProducer::close** ( ) throw (  
**cms::CMSException**) [virtual]

Closes the Consumer.

This will return all allocated resources and purge any outstanding messages. This method will block if there is a call to receive in progress, or a dispatch to a MessageListener in place

### Exceptions

<i>CMSException</i>
---------------------

Implements **cms::Closeable** (p. 1180).

**6.59.2.2** `virtual int activemq::core::ActiveMQProducer::getDeliveryMode ( ) const throw ( cms::CMSEException ) [inline, virtual]`

Gets the delivery mode for this Producer.

#### Returns

The DeliveryMode

Implements **cms::MessageProducer** (p. 2811).

**6.59.2.3** `virtual bool activemq::core::ActiveMQProducer::getDisableMessageID ( ) const throw ( cms::CMSEException ) [inline, virtual]`

Gets if Message Ids are disabled for this Producer.

#### Returns

a boolean indicating state enable / disable (true / false) for MessageIds.

Implements **cms::MessageProducer** (p. 2812).

**6.59.2.4** `virtual bool activemq::core::ActiveMQProducer::getDisableMessageTimeStamp ( ) const throw ( cms::CMSEException ) [inline, virtual]`

Gets if Message Time Stamps are disabled for this Producer.

#### Returns

boolean indicating state of enable / disable (true / false)

Implements **cms::MessageProducer** (p. 2812).

**6.59.2.5** `virtual int activemq::core::ActiveMQProducer::getPriority ( ) const throw ( cms::CMSEException ) [inline, virtual]`

Gets the Priority level that this producer sends messages at.

#### Returns

int based priority level

Implements **cms::MessageProducer** (p. 2812).

**6.59.2.6** `const Pointer<commands::ProducerId>& activemq::core::ActiveMQProducer::getProducerId ( ) const [inline]`

Retries this object ProducerId or NULL if closed.

**Returns**

ProducerId Reference

**6.59.2.7** `const Pointer<commands::ProducerInfo>&  
activemq::core::ActiveMQProducer::getProducerInfo ( ) const` [inline]

Retries this object ProducerInfo pointer.

**Returns**

ProducerInfo Reference

**6.59.2.8** `virtual long long activemq::core::ActiveMQProducer::getSendTimeout ( ) const throw (  
cms::CMSException )` [inline, virtual]

Gets the Send Timeout that this producer sends messages with.

**Returns**

The default send timeout value in milliseconds.

**6.59.2.9** `virtual long long activemq::core::ActiveMQProducer::getTimeToLive ( ) const throw (  
cms::CMSException )` [inline, virtual]

Gets the Time to Live that this producer sends messages with.

**Returns**

The default time to live value in milliseconds.

Implements **cms::MessageProducer** (p. 2813).

**6.59.2.10** `bool activemq::core::ActiveMQProducer::isClosed ( ) const` [inline]

**Returns**

true if this Producer has been closed.

**6.59.2.11** `virtual void activemq::core::ActiveMQProducer::onProducerAck ( const  
commands::ProducerAck & ack )` [virtual]

Handles the work of Processing a ProducerAck Command from the Broker.

**Parameters**

<i>ack</i>	- The ProducerAck message received from the Broker.
------------	---

**6.59.2.12** `virtual void activemq::core::ActiveMQProducer::send ( const cms::Destination * destination, cms::Message * message, int deliveryMode, int priority, long long timeToLive ) throw ( cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException ) [virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

#### Parameters

<i>destination</i>	The destination on which to send the message
<i>message</i>	The message to be sent.
<i>delivery-Mode</i>	The delivery mode to be used.
<i>priority</i>	The priority for this message.
<i>timeToLive</i>	The time to live value for this message in milliseconds.

#### Exceptions

<i>CMSEException</i>	- if an internal error occurs while sending the message.
<i>MessageFormatException</i>	- if an Invalid Message is given.
<i>InvalidDestinationException</i>	- if a client uses this method with a MessageProducer with an invalid destination.
<i>UnsupportedOperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements `cms::MessageProducer` (p. 2814).

**6.59.2.13** `virtual void activemq::core::ActiveMQProducer::send ( cms::Message * message ) throw ( cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException ) [virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

#### Parameters

<i>message</i>	The message to be sent.
----------------	-------------------------

**Exceptions**

<i>CMSEException</i>	- if an internal error occurs while sending the message.
<i>MessageFormatException</i>	- if an Invalid Message is given.
<i>InvalidDestinationException</i>	- if a client uses this method with a MessageProducer with an invalid destination.
<i>UnsupportedOperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2814).

**6.59.2.14** `virtual void activemq::core::ActiveMQProducer::send ( cms::Message * message, int deliveryMode, int priority, long long timeToLive ) throw ( cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException ) [virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

**Parameters**

<i>message</i>	The message to be sent.
<i>deliveryMode</i>	The delivery mode to be used.
<i>priority</i>	The priority for this message.
<i>timeToLive</i>	The time to live value for this message in milliseconds.

**Exceptions**

<i>CMSEException</i>	- if an internal error occurs while sending the message.
<i>MessageFormatException</i>	- if an Invalid Message is given.
<i>InvalidDestinationException</i>	- if a client uses this method with a MessageProducer with an invalid destination.
<i>UnsupportedOperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2813).

**6.59.2.15** `virtual void activemq::core::ActiveMQProducer::send ( const cms::Destination * destination, cms::Message * message ) throw ( cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException ) [virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.



**Parameters**

<i>destination</i>	The destination on which to send the message
<i>message</i>	the message to be sent.

**Exceptions**

<i>CMSException</i>	- if an internal error occurs while sending the message.
<i>MessageFormatException</i>	- if an Invalid Message is given.
<i>InvalidDestinationException</i>	- if a client uses this method with a MessageProducer with an invalid destination.
<i>UnsupportedOperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2815).

**6.59.2.16** `virtual void activemq::core::ActiveMQProducer::setDeliveryMode ( int mode ) throw ( cms::CMSException ) [inline, virtual]`

Sets the delivery mode for this Producer.

**Parameters**

<i>mode</i>	- The DeliveryMode to use for Message sends.
-------------	--

Implements **cms::MessageProducer** (p. 2816).

**6.59.2.17** `virtual void activemq::core::ActiveMQProducer::setDisableMessageID ( bool value ) throw ( cms::CMSException ) [inline, virtual]`

Sets if Message Ids are disabled for this Producer.

**Parameters**

<i>value</i>	- boolean indicating enable / disable (true / false)
--------------	--

Implements **cms::MessageProducer** (p. 2816).

**6.59.2.18** `virtual void activemq::core::ActiveMQProducer::setDisableMessageTimeStamp ( bool value ) throw ( cms::CMSException ) [inline, virtual]`

Sets if Message Time Stamps are disabled for this Producer.

**Parameters**

<i>value</i>	- boolean indicating enable / disable (true / false)
--------------	--

Implements **cms::MessageProducer** (p. 2816).

**6.59.2.19** `virtual void activemq::core::ActiveMQProducer::setPriority ( int priority ) throw ( cms::CMSException ) [inline, virtual]`

Sets the Priority that this Producers sends messages at.

#### Parameters

<i>priority</i>	int value for Priority level
-----------------	------------------------------

Implements **cms::MessageProducer** (p. 2817).

**6.59.2.20** `virtual void activemq::core::ActiveMQProducer::setSendTimeout ( long long time ) throw ( cms::CMSException ) [inline, virtual]`

Sets the Send Timeout that this Producers sends messages with.

#### Parameters

<i>time</i>	The new default send timeout value in milliseconds.
-------------	---

**6.59.2.21** `virtual void activemq::core::ActiveMQProducer::setTimeToLive ( long long time ) throw ( cms::CMSException ) [inline, virtual]`

Sets the Time to Live that this Producers sends messages with.

#### Parameters

<i>time</i>	The new default time to live value in milliseconds.
-------------	---

Implements **cms::MessageProducer** (p. 2817).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQProducer.h**

## 6.60 activemq::util::ActiveMQProperties Class Reference

Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 3216) object.

```
#include <src/main/activemq/util/ActiveMQProperties.h>
```

Inheritance diagram for activemq::util::ActiveMQProperties:

## Public Member Functions

- **ActiveMQProperties** ()
- virtual **~ActiveMQProperties** ()
- virtual **decaf::util::Properties** & **getProperties** ()
- virtual const **decaf::util::Properties** & **getProperties** () const
- virtual void **setProperty** (**decaf::util::Properties** &props)
- virtual bool **isEmpty** () const  
*Returns true if the properties object is empty.*
- virtual const char \* **getProperty** (const std::string &name) const  
*Looks up the value for the given property.*
- virtual std::string **getProperty** (const std::string &name, const std::string &defaultValue) const  
*Looks up the value for the given property.*
- virtual void **setProperty** (const std::string &name, const std::string &value)  
*Sets the value for a given property.*
- virtual bool **hasProperty** (const std::string &name) const  
*Check to see if the Property exists in the set.*
- virtual void **remove** (const std::string &name)  
*Removes the property with the given name.*
- virtual std::vector< std::pair< std::string, std::string > > **toArray** () const  
*Method that serializes the contents of the property map to an array.*
- virtual void **copy** (const CMSProperties \*source)  
*Copies the contents of the given properties object to this one.*
- virtual CMSProperties \* **clone** () const  
*Clones this object.*
- virtual void **clear** ()  
*Clears all properties from the map.*
- virtual std::string **toString** () const  
*Formats the contents of the Properties Object into a string that can be logged, etc.*

### 6.60.1 Detailed Description

Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 3216) object.

Since

2.0

### 6.60.2 Constructor & Destructor Documentation

6.60.2.1 **activemq::util::ActiveMQProperties::ActiveMQProperties ( )**

6.60.2.2 **virtual activemq::util::ActiveMQProperties::~~ActiveMQProperties ( )** [virtual]

### 6.60.3 Member Function Documentation

6.60.3.1 **virtual void activemq::util::ActiveMQProperties::clear ( )** [inline, virtual]

Clears all properties from the map.

Implements **cms::CMSProperties** (p. 1197).

6.60.3.2 **virtual CMSProperties\* activemq::util::ActiveMQProperties::clone ( )** const [virtual]

Clones this object.

**Returns**

a replica of this object.

Implements **cms::CMSProperties** (p. 1197).

6.60.3.3 **virtual void activemq::util::ActiveMQProperties::copy ( const CMSProperties \* source )** [virtual]

Copies the contents of the given properties object to this one.

**Parameters**

<i>source</i>	The source properties object.
---------------	-------------------------------

**6.60.3.4** `virtual decaf::util::Properties& activemq::util::ActiveMQProperties::getProperties ( ) [inline, virtual]`

**6.60.3.5** `virtual const decaf::util::Properties& activemq::util::ActiveMQProperties::getProperties ( ) const [inline, virtual]`

**6.60.3.6** `virtual const char* activemq::util::ActiveMQProperties::getProperty ( const std::string & name ) const [inline, virtual]`

Looks up the value for the given property.

#### Parameters

<i>name</i>	The name of the property to be looked up.
-------------	---

#### Returns

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

Implements **cms::CMSProperties** (p. 1197).

**6.60.3.7** `virtual std::string activemq::util::ActiveMQProperties::getProperty ( const std::string & name, const std::string & defaultValue ) const [inline, virtual]`

Looks up the value for the given property.

#### Parameters

<i>name</i>	the name of the property to be looked up.
<i>defaultValue</i>	The value to be returned if the given property does not exist.

#### Returns

The value of the property specified by name, if it exists, otherwise the `defaultValue`.

Implements **cms::CMSProperties** (p. 1198).

**6.60.3.8** `virtual bool activemq::util::ActiveMQProperties::hasProperty ( const std::string & name ) const [inline, virtual]`

Check to see if the Property exists in the set.

#### Parameters

<i>name</i>	- property name to check for in this properties set.
-------------	--

**Returns**

true if property exists, false otherwise.

Implements **cms::CMSProperties** (p. 1198).

**6.60.3.9** `virtual bool activemq::util::ActiveMQProperties::isEmpty ( ) const` `[inline, virtual]`

Returns true if the properties object is empty.

**Returns**

true if empty

Implements **cms::CMSProperties** (p. 1198).

**6.60.3.10** `virtual void activemq::util::ActiveMQProperties::remove ( const std::string & name )` `[inline, virtual]`

Removes the property with the given name.

**Parameters**

<i>name</i>	the name of the property to remove.
-------------	-------------------------------------

Implements **cms::CMSProperties** (p. 1198).

**6.60.3.11** `virtual void activemq::util::ActiveMQProperties::setProperties ( decaf::util::Properties & props )` `[inline, virtual]`

**6.60.3.12** `virtual void activemq::util::ActiveMQProperties::setProperty ( const std::string & name, const std::string & value )` `[inline, virtual]`

Sets the value for a given property.

If the property already exists, overwrites the value.

**Parameters**

<i>name</i>	The name of the value to be written.
<i>value</i>	The value to be written.

Implements **cms::CMSProperties** (p. 1199).

**6.60.3.13** `virtual std::vector< std::pair< std::string, std::string > >  
 activemq::util::ActiveMQProperties::toArray ( ) const [inline, virtual]`

Method that serializes the contents of the property map to an array.

#### Returns

list of pairs where the first is the name and the second is the value.

Implements **cms::CMSProperties** (p. 1199).

**6.60.3.14** `virtual std::string activemq::util::ActiveMQProperties::toString ( ) const  
 [inline, virtual]`

Formats the contents of the Properties Object into a string that can be logged, etc.

#### Returns

string value of this object.

Implements **cms::CMSProperties** (p. 1199).

The documentation for this class was generated from the following file:

- `src/main/activemq/util/ActiveMQProperties.h`

## 6.61 activemq::commands::ActiveMQQueue Class Reference

```
#include <src/main/activemq/commands/ActiveMQQueue.h>
```

Inheritance diagram for `activemq::commands::ActiveMQQueue`:

### Public Member Functions

- **ActiveMQQueue** ()
- **ActiveMQQueue** (const std::string &name)
- virtual **~ActiveMQQueue** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the **DataStructure** (p. 1713) Type as defined in **CommandTypes.h**.*
- virtual **ActiveMQQueue** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)

*Copy the contents of the passed object into this objects members, overwriting any existing data.*

- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const **cms::Destination** \* **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const  
*Retrieve the Destination Type for this Destination.*
- virtual **cms::Destination** \* **clone** () const  
*Creates a new instance of this destination type that is a copy of this one, and returns it.*
- virtual void **copy** (const **cms::Destination** &source)  
*Copies the contents of the given Destination object to this one.*
- virtual const **cms::CMSProperties** & **getCMSProperties** () const  
*Retrieve any properties that might be part of the destination that was specified.*
- virtual std::string **getQueueName** () const throw ( cms::CMSException )  
*Gets the name of this queue.*

## Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQUEUE** = 100

## 6.61.1 Constructor & Destructor Documentation

6.61.1.1 **activemq::commands::ActiveMQQueue::ActiveMQQueue ( )**

6.61.1.2 **activemq::commands::ActiveMQQueue::ActiveMQQueue ( const std::string & name )**

6.61.1.3 **virtual activemq::commands::ActiveMQQueue::~~ActiveMQQueue ( )** [inline, virtual]

## 6.61.2 Member Function Documentation

6.61.2.1 **virtual cms::Destination\* activemq::commands::ActiveMQQueue::clone ( )** const [inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.



**Returns**

cloned copy of this object

Implements **cms::Destination** (p. 1778).

**6.61.2.2** `virtual ActiveMQQueue* activemq::commands::ActiveMQQueue::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

**Returns**

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 315).

**6.61.2.3** `virtual void activemq::commands::ActiveMQQueue::copy ( const cms::Destination & source ) [inline, virtual]`

Copies the contents of the given Destination object to this one.

**Parameters**

<i>source</i>	The source Destination object.
---------------	--------------------------------

Implements **cms::Destination** (p. 1778).

**6.61.2.4** `virtual void activemq::commands::ActiveMQQueue::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

**Returns**

src - Source Object

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 316).

**6.61.2.5** `virtual bool activemq::commands::ActiveMQQueue::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 317).

```
6.61.2.6 virtual const cms::Destination* ac-
        tivemq::commands::ActiveMQQueue::getCMSDestination ( )
        const [inline, virtual]
```

**Returns**

the **cms::Destination** (p. 1776) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 317).

```
6.61.2.7 virtual const cms::CMSProperties& ac-
        tivemq::commands::ActiveMQQueue::getCMSProperties ( )
        const [inline, virtual]
```

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

**Returns**

const reference to a properties object.

Implements **cms::Destination** (p. 1778).

```
6.61.2.8 virtual unsigned char activemq::commands::ActiveMQQueue::getDataStructureType (
        ) const [virtual]
```

Get the **DataStructure** (p. 1713) Type as defined in CommandTypes.h.

**Returns**

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 317).

```
6.61.2.9 virtual cms::Destination::DestinationType
        activemq::commands::ActiveMQQueue::getDestinationType ( ) const [inline,
        virtual]
```

Retrieve the Destination Type for this Destination.

**Returns**

The Destination Type

Implements **activemq::commands::ActiveMQDestination** (p. 318).

References **cms::Destination::QUEUE**.

**6.61.2.10** `virtual std::string activemq::commands::ActiveMQQueue::getQueueName ( ) const  
throw ( cms::CMSException ) [inline, virtual]`

Gets the name of this queue.

#### Returns

The queue name.

Implements **cms::Queue** (p. 3239).

**6.61.2.11** `virtual std::string activemq::commands::ActiveMQQueue::toString ( ) const  
[virtual]`

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 321).

### 6.61.3 Field Documentation

**6.61.3.1** `const unsigned char activemq::commands::ActiveMQQueue::ID_-  
ACTIVEMQQUEUE = 100 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQQueue.h`

## 6.62 activemq::core::ActiveMQQueueBrowser Class Reference

```
#include <src/main/activemq/core/ActiveMQQueueBrowser.h>
```

Inheritance diagram for **activemq::core::ActiveMQQueueBrowser**:

## Public Member Functions

- **ActiveMQQueueBrowser** (**ActiveMQSession** \*session, const **Pointer**< **commands::ConsumerId** > &consumerId, const **Pointer**< **commands::ActiveMQDestination** > &destination, const std::string &selector, bool dispatchAsync)
- virtual ~**ActiveMQQueueBrowser** ()
- virtual const **cms::Queue** \* **getQueue** () const throw ( cms::CMSEException )
- virtual std::string **getMessageSelector** () const throw ( cms::CMSEException )
- virtual **cms::MessageEnumeration** \* **getEnumeration** () throw ( cms::CMSEException )

*Gets a pointer to an Enumeration object for browsing the Messages currently in the Queue in the order that a client would receive them.*

- virtual void **close** () throw ( cms::CMSEException )

*Closes this object and deallocates the appropriate resources.*

- virtual bool **hasMoreMessages** ()

*Returns true if there are more Message in the Browser that can be retrieved via the nextMessage method.*

- virtual **cms::Message** \* **nextMessage** () throw ( cms::CMSEException )

*Returns the Next Message in the Queue if one is present, if no more Message's are available then an Exception is thrown.*

## Friends

- class **Browser**

## 6.62.1 Constructor & Destructor Documentation

- 6.62.1.1 **activemq::core::ActiveMQQueueBrowser::ActiveMQQueueBrowser** (**ActiveMQSession** \* session, const **Pointer**< **commands::ConsumerId** > & consumerId, const **Pointer**< **commands::ActiveMQDestination** > & destination, const std::string & selector, bool dispatchAsync )

- 6.62.1.2 virtual **activemq::core::ActiveMQQueueBrowser::~~ActiveMQQueueBrowser** ( )  
[virtual]

## 6.62.2 Member Function Documentation

- 6.62.2.1 virtual void **activemq::core::ActiveMQQueueBrowser::close** ( ) throw ( **cms::CMSEException** ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

**Exceptions**

<i>CMSEException</i>	- If an error occurs while the resource is being closed.
----------------------	--

Implements **cms::Closeable** (p. 1180).

**6.62.2.2** `virtual cms::MessageEnumeration* activemq::core::ActiveMQQueueBrowser::getEnumeration ( ) throw ( cms::CMSEException ) [virtual]`

Gets a pointer to an Enumeration object for browsing the Messages currently in the Queue in the order that a client would receive them.

The pointer returned is owned by the browser and should not be deleted by the client application.

**Returns**

a pointer to a Queue Enumeration, this Pointer is owned by the QueueBrowser and should not be deleted by the client.

**Exceptions**

<i>CMSEException</i>	if an internal error occurs.
----------------------	------------------------------

Implements **cms::QueueBrowser** (p. 3244).

**6.62.2.3** `virtual std::string activemq::core::ActiveMQQueueBrowser::getMessageSelector ( ) const throw ( cms::CMSEException ) [virtual]`

**Returns**

the MessageSelector that is used on when this browser was created or empty string if no selector was present.

**Exceptions**

<i>CMSEException</i>	if an internal error occurs.
----------------------	------------------------------

Implements **cms::QueueBrowser** (p. 3245).

**6.62.2.4** `virtual const cms::Queue* activemq::core::ActiveMQQueueBrowser::getQueue ( ) const throw ( cms::CMSEException ) [virtual]`

**Returns**

the Queue that this browser is listening on.

**Exceptions**

<i>CMSEException</i>	if an internal error occurs.
----------------------	------------------------------

Implements **cms::QueueBrowser** (p. 3245).

**6.62.2.5** `virtual bool activemq::core::ActiveMQQueueBrowser::hasMoreMessages ( )`  
[virtual]

Returns true if there are more Message in the Browser that can be retrieved via the `nextMessage` method.

If this method returns false and the `nextMessage` method is called then an Exception will be thrown.

#### Returns

true if more Message's are available in the Browser.

Implements **cms::MessageEnumeration** (p. 2747).

**6.62.2.6** `virtual cms::Message* activemq::core::ActiveMQQueueBrowser::nextMessage ( )`  
`throw ( cms::CMSEException )` [virtual]

Returns the Next Message in the Queue if one is present, if no more Message's are available then an Exception is thrown.

If a Message object pointer is returned then that object becomes the property of the caller and must be deleted by the caller when finished.

#### Returns

The next Message in the Queue.

#### Exceptions

<i>CMSEException</i>	if no more Message's currently in the Queue.
----------------------	--

Implements **cms::MessageEnumeration** (p. 2747).

## 6.62.3 Friends And Related Function Documentation

**6.62.3.1** `friend class Browser` [friend]

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQQueueBrowser.h`

## 6.63 activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 487).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller:

### Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.63.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 487).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.63.2 Constructor & Destructor Documentation

6.63.2.1 **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller**  
 ( ) [inline]

6.63.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller**  
 ( ) [inline, virtual]

### 6.63.3 Member Function Documentation

6.63.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::createObject (**  
**) const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.63.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::getDataStructureType**  
**( ) const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.63.3.3 **virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure**  
**\* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw (**  
**decaf::io::IOException)** [virtual]

Write a object instance to data output stream.

#### Parameters



<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 325).

**6.63.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 325).

**6.63.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 326).

**6.63.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 326).

**6.63.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 327).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h

## 6.64 activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 491).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller**:

## Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.64.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 491).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.64.2 Constructor & Destructor Documentation

6.64.2.1 **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller**  
 ( ) [inline]

6.64.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller**  
 ( ) [inline, virtual]

### 6.64.3 Member Function Documentation

6.64.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.64.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

## 6.64 activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller

### Class Reference 493

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.64.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 329).

**6.64.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 329).

**6.64.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 330).

**6.64.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 330).

## 6.65 activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller

### Class Reference

495

6.64.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 331).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h

## 6.65 activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 495).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller**:

#### Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.65.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p.495).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.65.2 Constructor & Destructor Documentation

6.65.2.1 **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller**  
( ) [inline]

6.65.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller**  
( ) [inline, virtual]

### 6.65.3 Member Function Documentation

6.65.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.



### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.65.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.65.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 333).

**6.65.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 333).

```
6.65.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 334).

```
6.65.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink

## 6.66 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller

### Class Reference

499

<i>bs</i>	- BooleanStream stream used to pack bits from the wire.
-----------	---

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 334).

**6.65.3.7** virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 335).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h

## 6.66 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 499).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller:

## Public Member Functions

- **ActiveMQQueueMarshaller ()**
- virtual **~ActiveMQQueueMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn)** throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.66.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 499).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.66.2 Constructor & Destructor Documentation

6.66.2.1 **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller**  
( ) [inline]

6.66.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller**  
( ) [inline, virtual]

## 6.66.3 Member Function Documentation

6.66.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.66.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.66.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 337).

```
6.66.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 337).

```
6.66.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 338).

**6.66.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 338).

**6.66.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 339).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h`

## 6.67 activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 504).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMa
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller`:

### Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*



### 6.67.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 504).  
NOTE!: This file is auto generated - do not modify! if you need to make a change,  
please see the Java Classes in the activemq-openwire-generator module

### 6.67.2 Constructor & Destructor Documentation

**6.67.2.1** `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller  
( ) [inline]`

**6.67.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller  
( ) [inline, virtual]`

### 6.67.3 Member Function Documentation

**6.67.3.1** `virtual commands::DataStructure* ac-  
tivismq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.67.3.2** `virtual unsigned char activismq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::getDataStructureType  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.67.3.3** `virtual void activismq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::looseMarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 341).

```
6.67.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataInputStream * dataIn ) throw (
 decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 341).

```
6.67.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
 dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 342).

**6.67.3.6** virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::tightMarshal2  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure  
 \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
 utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 342).

**6.67.3.7** virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::tightUnmarshal  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \*  
*bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 343).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h`

## 6.68 `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller` Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 508).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQQueueMa
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller`:

**Public Member Functions**

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.68.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 508).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.68.2 Constructor & Destructor Documentation

6.68.2.1 **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller**  
( ) [inline]

6.68.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller**  
( ) [inline, virtual]

### 6.68.3 Member Function Documentation

6.68.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.68.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::getDataStructureType** ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.68.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 345).

**6.68.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 345).

6.68.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 346).

6.68.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 346).

```
6.68.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 347).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQQueueMarshaller.h`

## 6.69 activemq::core::ActiveMQSession Class Reference

```
#include <src/main/activemq/core/ActiveMQSession.h>
```

Inheritance diagram for `activemq::core::ActiveMQSession`:

#### Public Member Functions

- **ActiveMQSession** (const **Pointer**< **commands::SessionInfo** > &sessionInfo, **cms::Session::AcknowledgeMode** ackMode, const **decaf::util::Properties** &properties, **ActiveMQConnection** \*connection)
- virtual **~ActiveMQSession** ()
- void **redispatch** (**MessageDispatchChannel** &unconsumedMessages)  
*Redispatches the given set of unconsumed messages to the consumers.*
- void **start** ()  
*Stops asynchronous message delivery.*
- void **stop** ()  
*Starts asynchronous message delivery.*



- bool **isStarted** () const  
*Indicates whether or not the session is currently in the started state.*
- bool **isAutoAcknowledge** () const
- bool **isDupsOkAcknowledge** () const
- bool **isClientAcknowledge** () const
- bool **isIndividualAcknowledge** () const
- void **fire** (const **exceptions::ActiveMQException** &ex)  
*Fires the given exception to the exception listener of the connection.*
- virtual void **dispatch** (const **Pointer< MessageDispatch >** &message)  
*Dispatches a message to a particular consumer.*
- virtual void **close** () throw ( cms::CMSException )  
*Closes this session as well as any active child consumers or producers.*
- virtual void **commit** () throw ( cms::CMSException )  
*Commits all messages done in this transaction and releases any locks currently held.*
- virtual void **rollback** () throw ( cms::CMSException )  
*Rollsback all messages done in this transaction and releases any locks currently held.*
- virtual void **recover** () throw ( cms::CMSException )  
*Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.*
- virtual **cms::MessageConsumer** \* **createConsumer** (const **cms::Destination** \*destination) throw ( cms::CMSException )  
*Creates a MessageConsumer for the specified destination.*
- virtual **cms::MessageConsumer** \* **createConsumer** (const **cms::Destination** \*destination, const std::string &selector) throw ( cms::CMSException )  
*Creates a MessageConsumer for the specified destination, using a message selector.*
- virtual **cms::MessageConsumer** \* **createConsumer** (const **cms::Destination** \*destination, const std::string &selector, bool noLocal) throw ( cms::CMSException )  
*Creates a MessageConsumer for the specified destination, using a message selector.*
- virtual **cms::MessageConsumer** \* **createDurableConsumer** (const **cms::Topic** \*destination, const std::string &name, const std::string &selector, bool noLocal=false) throw ( cms::CMSException )  
*Creates a durable subscriber to the specified topic, using a message selector.*
- virtual **cms::MessageProducer** \* **createProducer** (const **cms::Destination** \*destination) throw ( cms::CMSException )

*Creates a MessageProducer to send messages to the specified destination.*

- virtual **cms::QueueBrowser \* createBrowser** (const **cms::Queue** \*queue) throw ( cms::CMSEException )

*Creates a new QueueBrowser to peek at Messages on the given Queue.*

- virtual **cms::QueueBrowser \* createBrowser** (const **cms::Queue** \*queue, const std::string &selector) throw ( cms::CMSEException )

*Creates a new QueueBrowser to peek at Messages on the given Queue.*

- virtual **cms::Queue \* createQueue** (const std::string &queueName) throw ( cms::CMSEException )

*Creates a queue identity given a Queue name.*

- virtual **cms::Topic \* createTopic** (const std::string &topicName) throw ( cms::CMSEException )

*Creates a topic identity given a Queue name.*

- virtual **cms::TemporaryQueue \* createTemporaryQueue** () throw ( cms::CMSEException )

*Creates a TemporaryQueue object.*

- virtual **cms::TemporaryTopic \* createTemporaryTopic** () throw ( cms::CMSEException )

*Creates a TemporaryTopic object.*

- virtual **cms::Message \* createMessage** () throw ( cms::CMSEException )

*Creates a new Message.*

- virtual **cms::BytesMessage \* createBytesMessage** () throw ( cms::CMSEException )

*Creates a BytesMessage.*

- virtual **cms::BytesMessage \* createBytesMessage** (const unsigned char \*bytes, int bytesSize) throw ( cms::CMSEException )

*Creates a BytesMessage and sets the pay-load to the passed value.*

- virtual **cms::StreamMessage \* createStreamMessage** () throw ( cms::CMSEException )

*Creates a new StreamMessage.*

- virtual **cms::TextMessage \* createTextMessage** () throw ( cms::CMSEException )

*Creates a new TextMessage.*

- virtual **cms::TextMessage \* createTextMessage** (const std::string &text) throw ( cms::CMSEException )

*Creates a new TextMessage and set the text to the value given.*

- virtual **cms::MapMessage \* createMapMessage ()** throw ( cms::CMSEException )

*Creates a new MapMessage.*

- virtual **cms::Session::AcknowledgeMode getAcknowledgeMode ()** const throw ( cms::CMSEException )

*Returns the acknowledgment mode of the session.*

- virtual bool **isTransacted ()** const throw ( cms::CMSEException )

*Gets if the Sessions is a Transacted Session.*

- virtual void **unsubscribe** (const std::string &name) throw ( cms::CMSEException )

*Unsubscribes a durable subscription that has been created by a client.*

- void **send (cms::Message \*message, ActiveMQProducer \*producer, util::Usage \*usage)** throw ( cms::CMSEException )

*Sends a message from the Producer specified using this session's connection the message will be sent using the best available means depending on the configuration of the connection.*

- **cms::ExceptionListener \* getExceptionListener ()**

*This method gets any registered exception listener of this sessions connection and returns it.*

- const **commands::SessionInfo & getSessionInfo ()** const

*Gets the Session Information object for this session, if the session is closed than this method throws an exception.*

- const **commands::SessionId & getSessionId ()** const

*Gets the Session Id object for this session, if the session is closed than this method throws an exception.*

- **ActiveMQConnection \* getConnection ()** const

*Gets the ActiveMQConnection (p. 260) that is associated with this session.*

- long long **getLastDeliveredSequenceId ()** const

*Gets the currently set Last Delivered Sequence Id.*

- void **setLastDeliveredSequenceId** (long long value)

*Sets the value of the Last Delivered Sequence Id.*

- void **oneway (Pointer< commands::Command > command)** throw ( activemq::exceptions::ActiveMQException )

*Sends a oneway message.*

- **void syncRequest (Pointer< commands::Command > command, unsigned int timeout=0) throw ( activemq::exceptions::ActiveMQException )**  
*Sends a synchronous request and returns the response from the broker.*
- **void addConsumer (ActiveMQConsumer \*consumer) throw ( activemq::exceptions::ActiveMQException )**  
*Adds a MessageConsumer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.*
- **void removeConsumer (const Pointer< commands::ConsumerId > &consumerId, long long lastDeliveredSequenceId=0) throw ( activemq::exceptions::ActiveMQException )**  
*Dispose of a MessageConsumer from this session.*
- **void addProducer (ActiveMQProducer \*consumer) throw ( activemq::exceptions::ActiveMQException )**  
*Adds a MessageProducer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.*
- **void removeProducer (const Pointer< commands::ProducerId > &producerId) throw ( activemq::exceptions::ActiveMQException )**  
*Dispose of a MessageProducer from this session.*
- **void doStartTransaction () throw ( exceptions::ActiveMQException )**  
*Starts if not already start a Transaction for this Session.*
- **Pointer< ActiveMQTransactionContext > getTransactionContext ()**  
*Gets the Pointer to this Session's TransactionContext.*
- **void acknowledge ()**  
*Request that the Session inform all its consumers to Acknowledge all Message's that have been received so far.*
- **void deliverAcks ()**  
*Request that this Session inform all of its consumers to deliver their pending acks.*
- **void clearMessagesInProgress ()**  
*Request that this Session inform all of its consumers to clear all messages that are currently in progress.*
- **void wakeup ()**  
*Causes the Session to wakeup its executer and ensure all messages are dispatched.*
- **Pointer< commands::ConsumerId > getNextConsumerId ()**

*Get the Next available Consumer Id.*

- **Pointer< commands::ProducerId > getNextProducerId ()**

*Get the Next available Producer Id.*

## Friends

- class **ActiveMQSessionExecutor**

## 6.69.1 Constructor & Destructor Documentation

**6.69.1.1** `activemq::core::ActiveMQSession::ActiveMQSession ( const Pointer< commands::SessionInfo > & sessionInfo, cms::Session::AcknowledgeMode ackMode, const decaf::util::Properties & properties, ActiveMQConnection * connection )`

**6.69.1.2** `virtual activemq::core::ActiveMQSession::~~ActiveMQSession ( )` [virtual]

## 6.69.2 Member Function Documentation

**6.69.2.1** `void activemq::core::ActiveMQSession::acknowledge ( )`

Request that the Session inform all its consumers to Acknowledge all Message's that have been received so far.

**6.69.2.2** `void activemq::core::ActiveMQSession::addConsumer ( ActiveMQConsumer * consumer ) throw ( activemq::exceptions::ActiveMQException )`

Adds a MessageConsumer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.

### Parameters

<i>consumer</i>	The <b>ActiveMQConsumer</b> (p. 300) instance to add to this session.
-----------------	---

### Exceptions

<i>ActiveMQException</i>	if an internal error occurs.
--------------------------	------------------------------

**6.69.2.3** `void activemq::core::ActiveMQSession::addProducer ( ActiveMQProducer * consumer ) throw ( activemq::exceptions::ActiveMQException )`

Adds a MessageProducer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session

is closed.

#### Parameters

<i>consumer</i>	The <b>ActiveMQProducer</b> (p. 466) instance to add to this session.
-----------------	---

#### Exceptions

<i>ActiveMQException</i>	if an internal error occurs.
--------------------------	------------------------------

#### 6.69.2.4 void activemq::core::ActiveMQSession::clearMessagesInProgress ( )

Request that this Session inform all of its consumers to clear all messages that are currently in progress.

#### 6.69.2.5 virtual void activemq::core::ActiveMQSession::close ( ) throw ( cms::CMSException ) [virtual]

Closes this session as well as any active child consumers or producers.

#### Exceptions

<i>CMSException</i>	
---------------------	--

Implements **cms::Session** (p. 3464).

#### 6.69.2.6 virtual void activemq::core::ActiveMQSession::commit ( ) throw ( cms::CMSException ) [virtual]

Commits all messages done in this transaction and releases any locks currently held.

#### Exceptions

<i>CMSException</i>	
---------------------	--

Implements **cms::Session** (p. 3465).

#### 6.69.2.7 virtual cms::QueueBrowser\* activemq::core::ActiveMQSession::createBrowser ( const cms::Queue \* queue ) throw ( cms::CMSException ) [virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

#### Parameters

<i>queue</i>	the Queue to browse
--------------	---------------------

**Returns**

New QueueBrowser that is owned by the caller.

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
<i>InvalidDestinationException</i>	- if the destination given is invalid.

Implements **cms::Session** (p. 3465).

**6.69.2.8** `virtual cms::QueueBrowser* activemq::core::ActiveMQSession::createBrowser ( const cms::Queue * queue, const std::string & selector ) throw ( cms::CMSEException )` [virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

**Parameters**

<i>queue</i>	the Queue to browse
<i>selector</i>	the Message selector to filter which messages are browsed.

**Returns**

New QueueBrowser that is owned by the caller.

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
<i>InvalidDestinationException</i>	- if the destination given is invalid.

Implements **cms::Session** (p. 3466).

**6.69.2.9** `virtual cms::BytesMessage* activemq::core::ActiveMQSession::createBytesMessage ( ) throw ( cms::CMSEException )` [virtual]

Creates a BytesMessage.

**Returns**

a newly created BytesMessage.

**Exceptions**

<i>CMSEException</i>	
----------------------	--

Implements **cms::Session** (p. 3466).

**6.69.2.10** `virtual cms::BytesMessage* activemq::core::ActiveMQSession::createBytesMessage ( const unsigned char * bytes, int bytesSize ) throw ( cms::CMSEException )`  
[virtual]

Creates a BytesMessage and sets the pay-load to the passed value.

#### Parameters

<i>bytes</i>	- an array of bytes to set in the message
<i>bytesSize</i>	- the size of the bytes array, or number of bytes to use

#### Returns

a newly created BytesMessage.

#### Exceptions

<i>CMSEException</i>
----------------------

Implements **cms::Session** (p. 3466).

**6.69.2.11** `virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer ( const cms::Destination * destination, const std::string & selector, bool noLocal ) throw ( cms::CMSEException )`  
[virtual]

Creates a MessageConsumer for the specified destination, using a message selector.

#### Parameters

<i>destination</i>	- The Destination that this consumer receiving messages for.
<i>selector</i>	- The Message Selector string to use for this destination
<i>noLocal</i>	- if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

#### Exceptions

<i>CMSEException</i>
----------------------

Implements **cms::Session** (p. 3468).

**6.69.2.12** `virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer ( const cms::Destination * destination, const std::string & selector ) throw ( cms::CMSEException )`  
[virtual]

Creates a MessageConsumer for the specified destination, using a message selector.



**Parameters**

<i>destination</i>	- The Destination that this consumer receiving messages for.
<i>selector</i>	- The Message Selector string to use for this destination

**Exceptions**

<i>CMSEException</i>	
----------------------	--

Implements **cms::Session** (p. 3467).

**6.69.2.13** `virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer ( const cms::Destination * destination ) throw ( cms::CMSEException )` [virtual]

Creates a MessageConsumer for the specified destination.

**Parameters**

<i>destination</i>	- The Destination that this consumer receiving messages for.
--------------------	--

**Exceptions**

<i>CMSEException</i>	
----------------------	--

Implements **cms::Session** (p. 3467).

**6.69.2.14** `virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createDurableConsumer ( const cms::Topic * destination, const std::string & name, const std::string & selector, bool noLocal = false ) throw ( cms::CMSEException )` [virtual]

Creates a durable subscriber to the specified topic, using a message selector.

**Parameters**

<i>destination</i>	- the topic to subscribe to
<i>name</i>	- The name used to identify the subscription
<i>selector</i>	- only messages matching the selector are received
<i>noLocal</i>	- if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

**Exceptions**

<i>CMSEException</i>	
----------------------	--

Implements **cms::Session** (p. 3469).

**6.69.2.15** `virtual cms::MapMessage* activemq::core::ActiveMQSession::createMapMessage ( ) throw ( cms::CMSEException ) [virtual]`

Creates a new MapMessage.

#### Returns

a newly created MapMessage.

#### Exceptions

<i>CMSEException</i>
----------------------

Implements **cms::Session** (p. 3469).

**6.69.2.16** `virtual cms::Message* activemq::core::ActiveMQSession::createMessage ( ) throw ( cms::CMSEException ) [virtual]`

Creates a new Message.

#### Exceptions

<i>CMSEException</i>
----------------------

Implements **cms::Session** (p. 3470).

**6.69.2.17** `virtual cms::MessageProducer* activemq::core::ActiveMQSession::createProducer ( const cms::Destination * destination ) throw ( cms::CMSEException ) [virtual]`

Creates a MessageProducer to send messages to the specified destination.

#### Parameters

<i>destination</i>	- the Destination to publish on
--------------------	---------------------------------

#### Exceptions

<i>CMSEException</i>
----------------------

Implements **cms::Session** (p. 3470).

**6.69.2.18** `virtual cms::Queue* activemq::core::ActiveMQSession::createQueue ( const std::string & queueName ) throw ( cms::CMSEException ) [virtual]`

Creates a queue identity given a Queue name.

**Parameters**

<i>queueName</i>	- the name of the new Queue
------------------	-----------------------------

**Exceptions**

<i>CMSEException</i>	
----------------------	--

Implements **cms::Session** (p. 3470).

**6.69.2.19** virtual **cms::StreamMessage\*** **activemq::core::ActiveMQSession::createStreamMessage** ( )  
throw ( **cms::CMSEException** ) [virtual]

Creates a new StreamMessage.

**Returns**

a newly created StreamMessage.

**Exceptions**

<i>CMSEException</i>	
----------------------	--

Implements **cms::Session** (p. 3471).

**6.69.2.20** virtual **cms::TemporaryQueue\*** **activemq::core::ActiveMQSession::createTemporaryQueue** ( )  
throw ( **cms::CMSEException** ) [virtual]

Creates a TemporaryQueue object.

**Exceptions**

<i>CMSEException</i>	
----------------------	--

Implements **cms::Session** (p. 3471).

**6.69.2.21** virtual **cms::TemporaryTopic\*** **activemq::core::ActiveMQSession::createTemporaryTopic** ( )  
throw ( **cms::CMSEException** ) [virtual]

Creates a TemporaryTopic object.

**Exceptions**

<i>CMSEException</i>	
----------------------	--

Implements **cms::Session** (p. 3471).

**6.69.2.22** `virtual cms::TextMessage* activemq::core::ActiveMQSession::createTextMessage  
( ) throw ( cms::CMSException ) [virtual]`

Creates a new TextMessage.

#### Returns

a newly created TextMessage.

#### Exceptions

<i>CMSException</i>
---------------------

Implements **cms::Session** (p. 3472).

**6.69.2.23** `virtual cms::TextMessage* activemq::core::ActiveMQSession::createTextMessage  
( const std::string & text ) throw ( cms::CMSException ) [virtual]`

Creates a new TextMessage and set the text to the value given.

#### Parameters

<i>text</i>	- The initial text for the message
-------------	------------------------------------

#### Returns

a newly created TextMessage with the given Text set in the Message body.

#### Exceptions

<i>CMSException</i>
---------------------

Implements **cms::Session** (p. 3472).

**6.69.2.24** `virtual cms::Topic* activemq::core::ActiveMQSession::createTopic ( const  
std::string & topicName ) throw ( cms::CMSException ) [virtual]`

Creates a topic identity given a Queue name.

#### Parameters

<i>topicName</i>	- the name of the new Topic
------------------	-----------------------------

#### Exceptions

<i>CMSException</i>
---------------------

Implements **cms::Session** (p. 3472).

**6.69.2.25 void activemq::core::ActiveMQSession::deliverAcks ( )**

Request that this Session inform all of its consumers to deliver their pending acks.

**6.69.2.26 virtual void activemq::core::ActiveMQSession::dispatch ( const Pointer< MessageDispatch > & message ) [virtual]**

Dispatches a message to a particular consumer.

**Parameters**

<i>message</i>	- the message to be dispatched
----------------	--------------------------------

Implements **activemq::core::Dispatcher** (p. 1841).

**6.69.2.27 void activemq::core::ActiveMQSession::doStartTransaction ( ) throw ( exceptions::ActiveMQException )**

Starts if not already start a Transaction for this Session.

If the session is not a Transacted Session then an exception is thrown. If a transaction is already in progress then this method has no effect.

**Exceptions**

<i>ActiveMQException</i>	if this is not a Transacted Session.
--------------------------	--------------------------------------

**6.69.2.28 void activemq::core::ActiveMQSession::fire ( const exceptions::ActiveMQException & ex )**

Fires the given exception to the exception listener of the connection.

**6.69.2.29 virtual cms::Session::AcknowledgeMode activemq::core::ActiveMQSession::getAcknowledgeMode ( ) const throw ( cms::CMSException ) [virtual]**

Returns the acknowledgment mode of the session.

**Returns**

the Sessions Acknowledge Mode

Implements **cms::Session** (p. 3473).

**6.69.2.30** `ActiveMQConnection* activemq::core::ActiveMQSession::getConnection ( )`  
`const [inline]`

Gets the **ActiveMQConnection** (p. 260) that is associated with this session.

**6.69.2.31** `cms::ExceptionListener* activemq::core::ActiveMQSession::getExceptionListener ( )`

This method gets any registered exception listener of this sessions connection and returns it.

Mainly intended for use by the objects that this session creates so that they can notify the client of exceptions that occur in the context of another thread.

#### Returns

`cms::ExceptionListener` (p. 1893) pointer or NULL

**6.69.2.32** `long long activemq::core::ActiveMQSession::getLastDeliveredSequenceId ( ) const`  
`[inline]`

Gets the currently set Last Delivered Sequence Id.

#### Returns

long long containing the sequence id of the last delivered Message.

**6.69.2.33** `Pointer<commands::ConsumerId> activemq::core::ActiveMQSession::getNextConsumerId ( )`

Get the Next available Consumer Id.

#### Returns

the next id in the sequence.

**6.69.2.34** `Pointer<commands::ProducerId> activemq::core::ActiveMQSession::getNextProducerId ( )`

Get the Next available Producer Id.

#### Returns

the next id in the sequence.

**6.69.2.35** `const commands::SessionId& activemq::core::ActiveMQSession::getSessionId ( ) const [inline]`

Gets the Session Id object for this session, if the session is closed than this method throws an exception.

#### Returns

SessionId Reference

**6.69.2.36** `const commands::SessionInfo& activemq::core::ActiveMQSession::getSessionInfo ( ) const [inline]`

Gets the Session Information object for this session, if the session is closed than this method throws an exception.

#### Returns

SessionInfo Reference

**6.69.2.37** `Pointer<ActiveMQTransactionContext> activemq::core::ActiveMQSession::getTransactionContext ( ) [inline]`

Gets the Pointer to this Session's TransactionContext.

#### Returns

a Pointer to this Session's TransactionContext

**6.69.2.38** `bool activemq::core::ActiveMQSession::isAutoAcknowledge ( ) const [inline]`

References cms::Session::AUTO\_ACKNOWLEDGE.

**6.69.2.39** `bool activemq::core::ActiveMQSession::isClientAcknowledge ( ) const [inline]`

References cms::Session::CLIENT\_ACKNOWLEDGE.

**6.69.2.40** `bool activemq::core::ActiveMQSession::isDupsOkAcknowledge ( ) const [inline]`

References cms::Session::DUPS\_OK\_ACKNOWLEDGE.

**6.69.2.41** `bool activemq::core::ActiveMQSession::isIndividualAcknowledge ( ) const`  
`[inline]`

References `cms::Session::INDIVIDUAL_ACKNOWLEDGE`.

**6.69.2.42** `bool activemq::core::ActiveMQSession::isStarted ( ) const`

Indicates whether or not the session is currently in the started state.

**6.69.2.43** `virtual bool activemq::core::ActiveMQSession::isTransacted ( ) const throw (`  
`cms::CMSException ) [virtual]`

Gets if the Sessions is a Transacted Session.

### Returns

transacted true - false.

Implements `cms::Session` (p. 3473).

**6.69.2.44** `void activemq::core::ActiveMQSession::oneway ( Pointer<`  
`commands::Command > command ) throw (`  
`activemq::exceptions::ActiveMQException )`

Sends a oneway message.

### Parameters

<i>command</i>	The message to send.
----------------	----------------------

### Exceptions

<i>ActiveMQException</i>	if not currently connected, or if the operation fails for any reason.
--------------------------	---

**6.69.2.45** `virtual void activemq::core::ActiveMQSession::recover ( ) throw (`  
`cms::CMSException ) [virtual]`

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.

All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "re-delivered"



- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

### Exceptions

<i>CMSException</i>	- if the CMS provider fails to stop and restart message delivery due to some internal error.
<i>IllegalStateException</i>	- if the method is called by a transacted session.

Implements **cms::Session** (p. 3474).

#### 6.69.2.46 void activemq::core::ActiveMQSession::redispatch ( MessageDispatchChannel & unconsumedMessages )

Redispatches the given set of unconsumed messages to the consumers.

### Parameters

<i>unconsumedMessages</i>	- unconsumed messages to be redelivered.
---------------------------	--

#### 6.69.2.47 void activemq::core::ActiveMQSession::removeConsumer ( const Pointer< commands::ConsumerId > & consumerId, long long lastDeliveredSequenceId = 0 ) throw ( activemq::exceptions::ActiveMQException )

Dispose of a MessageConsumer from this session.

Removes it from the Connection and clean up any resources associated with it.

### Parameters

<i>consumerId</i>	The ConsumerId of the MessageConsumer to remove from this Session.
<i>lastDeliveredSequenceId</i>	The sequenceId of the last Message the consumer delivered.

### Exceptions

<i>ActiveMQException</i>	if an internal error occurs.
--------------------------	------------------------------

#### 6.69.2.48 void activemq::core::ActiveMQSession::removeProducer ( const Pointer< commands::ProducerId > & producerId ) throw ( activemq::exceptions::ActiveMQException )

Dispose of a MessageProducer from this session.

Removes it from the Connection and clean up any resources associated with it.

#### Parameters

<i>producerId</i>	The ProducerId of the MessageProducer to remove from this session.
-------------------	--

#### Exceptions

<i>ActiveMQException</i>	if an internal error occurs.
--------------------------	------------------------------

**6.69.2.49** `virtual void activemq::core::ActiveMQSession::rollback ( ) throw ( cms::CMSException )` [virtual]

Rollsback all messages done in this transaction and releases any locks currently held.

#### Exceptions

<i>CMSException</i>	
---------------------	--

Implements **cms::Session** (p. 3474).

**6.69.2.50** `void activemq::core::ActiveMQSession::send ( cms::Message * message, ActiveMQProducer * producer, util::Usage * usage ) throw ( cms::CMSException )`

Sends a message from the Producer specified using this session's connection the message will be sent using the best available means depending on the configuration of the connection.

Asynchronous sends will be chosen if at all possible.

#### Parameters

<i>message</i>	The message to send to the broker.
<i>producer</i>	The sending Producer
<i>usage</i>	Pointer to a Usage tracker which if set will be increased by the size of the given message.

#### Exceptions

<i>CMSException</i>	
---------------------	--

**6.69.2.51** `void activemq::core::ActiveMQSession::setLastDeliveredSequenceId ( long long value )` [inline]

Sets the value of the Last Delivered Sequence Id.

**Parameters**

<i>value</i>	The new value to assign to the Last Delivered Sequence Id property.
--------------	---

**6.69.2.52 void activemq::core::ActiveMQSession::start ( )**

Stops asynchronous message delivery.

**6.69.2.53 void activemq::core::ActiveMQSession::stop ( )**

Starts asynchronous message delivery.

**6.69.2.54 void activemq::core::ActiveMQSession::syncRequest ( **Pointer**< **commands::Command** > *command*, unsigned int *timeout* = 0 ) throw ( **activemq::exceptions::ActiveMQException** )**

Sends a synchronous request and returns the response from the broker.

Converts any error responses into an exception.

**Parameters**

<i>command</i>	The request command.
<i>timeout</i>	The time to wait for a response, default is zero or infinite.

**Exceptions**

<i>ActiveMQException</i>	thrown if an error response was received from the broker, or if any other error occurred.
--------------------------	---

**6.69.2.55 virtual void activemq::core::ActiveMQSession::unsubscribe ( const std::string & *name* ) throw ( **cms::CMSException** ) [virtual]**

Unsubscribes a durable subscription that has been created by a client.

This method deletes the state being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active MessageConsumer or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

**Parameters**

<i>name</i>	the name used to identify this subscription
-------------	---

**Exceptions**

<i>CMSException</i>	
---------------------	--

Implements **cms::Session** (p. 3475).

### 6.69.2.56 void activemq::core::ActiveMQSession::wakeup ( )

Causes the Session to wakeup its executor and ensure all messages are dispatched.

## 6.69.3 Friends And Related Function Documentation

### 6.69.3.1 friend class ActiveMQSessionExecutor [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/core/ActiveMQSession.h

## 6.70 activemq::core::ActiveMQSessionExecutor Class Reference

Delegate dispatcher for a single session.

```
#include <src/main/activemq/core/ActiveMQSessionExecutor.h>
```

Inheritance diagram for activemq::core::ActiveMQSessionExecutor:

### Public Member Functions

- **ActiveMQSessionExecutor** (**ActiveMQSession** \*session)  
*Creates an un-started executor for the given session.*
- virtual **~ActiveMQSessionExecutor** ()  
*Calls **stop()** (p. 535) then **clear()** (p. 533).*
- virtual void **execute** (const **Pointer**< **MessageDispatch** > &data)  
*Executes the dispatch.*
- virtual void **executeFirst** (const **Pointer**< **MessageDispatch** > &data)  
*Executes the dispatch.*
- virtual void **clearMessagesInProgress** ()  
*Removes all messages in the Dispatch Channel so that non are delivered.*
- virtual bool **hasUnconsumedMessages** () const
- virtual void **wakeup** ()  
*wakeup this executor and dispatch any pending messages.*
- virtual void **start** ()  
*Starts the dispatching.*

- virtual void **stop** ()  
*Stops dispatching.*
- virtual void **close** ()  
*Terminates the dispatching thread.*
- virtual bool **isRunning** () const
- virtual bool **isEmpty** ()
- virtual void **clear** ()  
*Removes all queued messages and destroys them.*
- virtual bool **iterate** ()  
*Iterates on the **MessageDispatchChannel** (p. 2683) sending all pending messages to the Consumers they are destined for.*
- std::vector< **Pointer**< **MessageDispatch** > > **getUnconsumedMessages** ()

### 6.70.1 Detailed Description

Delegate dispatcher for a single session. Contains a thread to provide for asynchronous dispatching.

### 6.70.2 Constructor & Destructor Documentation

#### 6.70.2.1 **activemq::core::ActiveMQSessionExecutor::ActiveMQSessionExecutor ( **ActiveMQSession** \* *session* )**

Creates an un-started executor for the given session.

#### 6.70.2.2 **virtual activemq::core::ActiveMQSessionExecutor::~~ActiveMQSessionExecutor ( )** [virtual]

Calls **stop**() (p. 535) then **clear**() (p. 533).

### 6.70.3 Member Function Documentation

#### 6.70.3.1 **virtual void activemq::core::ActiveMQSessionExecutor::clear ( )** [inline, virtual]

Removes all queued messages and destroys them.

#### 6.70.3.2 **virtual void activemq::core::ActiveMQSessionExecutor::clearMessagesInProgress ( )** [inline, virtual]

Removes all messages in the Dispatch Channel so that non are delivered.

**6.70.3.3** `virtual void activemq::core::ActiveMQSessionExecutor::close ( )` [inline, virtual]

Terminates the dispatching thread.

Once this is called, the executor is no longer usable.

**6.70.3.4** `virtual void activemq::core::ActiveMQSessionExecutor::execute ( const Pointer< MessageDispatch > & data )` [virtual]

Executes the dispatch.

Adds the given data to the end of the queue.

#### Parameters

<i>data</i>	- the data to be dispatched.
-------------	------------------------------

**6.70.3.5** `virtual void activemq::core::ActiveMQSessionExecutor::executeFirst ( const Pointer< MessageDispatch > & data )` [virtual]

Executes the dispatch.

Adds the given data to the beginning of the queue.

#### Parameters

<i>data</i>	- the data to be dispatched.
-------------	------------------------------

**6.70.3.6** `std::vector< Pointer<MessageDispatch> > activemq::core::ActiveMQSessionExecutor::getUnconsumedMessages ( )` [inline]

#### Returns

a vector containing all the unconsumed messages, this clears the Message Dispatch Channel when called.

**6.70.3.7** `virtual bool activemq::core::ActiveMQSessionExecutor::hasUnconsumedMessages ( ) const` [inline, virtual]

#### Returns

true if there are any pending messages in the dispatch channel.

**6.70.3.8** `virtual bool activemq::core::ActiveMQSessionExecutor::isEmpty ( ) [inline, virtual]`

#### Returns

true if there are no messages in the Dispatch Channel.

**6.70.3.9** `virtual bool activemq::core::ActiveMQSessionExecutor::isRunning ( ) const [inline, virtual]`

#### Returns

true indicates if the executor is started

**6.70.3.10** `virtual bool activemq::core::ActiveMQSessionExecutor::iterate ( ) [virtual]`

Iterates on the **MessageDispatchChannel** (p. 2683) sending all pending messages to the Consumers they are destined for.

#### Returns

false if there are no more messages to dispatch.

Implements **activemq::threads::Task** (p. 3847).

**6.70.3.11** `virtual void activemq::core::ActiveMQSessionExecutor::start ( ) [virtual]`

Starts the dispatching.

**6.70.3.12** `virtual void activemq::core::ActiveMQSessionExecutor::stop ( ) [virtual]`

Stops dispatching.

**6.70.3.13** `virtual void activemq::core::ActiveMQSessionExecutor::wakeup ( ) [virtual]`

wakeup this executor and dispatch any pending messages.

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQSessionExecutor.h`

## 6.71 activemq::commands::ActiveMQStreamMessage Class Reference

```
#include <src/main/activemq/commands/ActiveMQStreamMessage.h>
```

Inheritance diagram for `activemq::commands::ActiveMQStreamMessage`:

## Public Member Functions

- **ActiveMQStreamMessage** ()
- virtual **~ActiveMQStreamMessage** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ActiveMQStreamMessage \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual void **onSend** ()  
*Store the Data that was written to the stream before a send.*
- virtual **cms::StreamMessage \* clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*
- virtual void **clearBody** () throw ( cms::CMSException )  
*Clears out the body of the message.*
- virtual void **reset** () throw ( cms::CMSException )  
*Puts the message body in read-only mode and repositions the stream of bytes to the beginning.*
- virtual bool **readBoolean** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException )  
*Reads a Boolean from the Stream message stream.*
- virtual void **writeBoolean** (bool value) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Writes a boolean to the Stream message stream as a 1-byte value.*



- virtual unsigned char **readByte** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Byte from the Stream message stream.*
- virtual void **writeByte** (unsigned char value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a byte to the Stream message stream as a 1-byte value.*
- virtual int **readBytes** (std::vector< unsigned char > &value) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a byte array from the Stream message stream.*
- virtual void **writeBytes** (const std::vector< unsigned char > &value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.*
- virtual int **readBytes** (unsigned char \*buffer, int length) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a portion of the Stream message stream.*
- virtual void **writeBytes** (const unsigned char \*value, int offset, int length) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a portion of a byte array to the Stream message stream.*
- virtual char **readChar** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Char from the Stream message stream.*
- virtual void **writeChar** (char value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a char to the Stream message stream as a 1-byte value.*
- virtual float **readFloat** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 32 bit float from the Stream message stream.*
- virtual void **writeFloat** (float value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a float to the Stream message stream as a 4 byte value.*
- virtual double **readDouble** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 64 bit double from the Stream message stream.*

- virtual void **writeDouble** (double value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a double to the Stream message stream as a 8 byte value.*

- virtual short **readShort** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 16 bit signed short from the Stream message stream.*

- virtual void **writeShort** (short value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a signed short to the Stream message stream as a 2 byte value.*

- virtual unsigned short **readUnsignedShort** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 16 bit unsigned short from the Stream message stream.*

- virtual void **writeUnsignedShort** (unsigned short value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a unsigned short to the Stream message stream as a 2 byte value.*

- virtual int **readInt** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 32 bit signed integer from the Stream message stream.*

- virtual void **writeInt** (int value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a signed int to the Stream message stream as a 4 byte value.*

- virtual long long **readLong** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 64 bit long from the Stream message stream.*

- virtual void **writeLong** (long long value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a long long to the Stream message stream as a 8 byte value.*

- virtual std::string **readString** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads an ASCII String from the Stream message stream.*

- virtual void **writeString** (const std::string &value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes an ASCII String to the Stream message stream.*

## Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQSTREAMMESSAGE** = 27

### 6.71.1 Constructor & Destructor Documentation

6.71.1.1 **activemq::commands::ActiveMQStreamMessage::ActiveMQStreamMessage ( )**

6.71.1.2 **virtual activemq::commands::ActiveMQStreamMessage::~~ActiveMQStreamMessage ( )** [virtual]

### 6.71.2 Member Function Documentation

6.71.2.1 **virtual void activemq::commands::ActiveMQStreamMessage::clearBody ( )** throw ( cms::CMSException ) [virtual]

Clears out the body of the message.

This does not clear the headers or properties.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 422).

6.71.2.2 **virtual cms::StreamMessage\* activemq::commands::ActiveMQStreamMessage::clone ( )** const [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

#### Returns

new copy of this message

Implements **cms::Message** (p. 2620).

6.71.2.3 **virtual ActiveMQStreamMessage\* activemq::commands::ActiveMQStreamMessage::cloneDataStructure ( )** const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 2601).

**6.71.2.4** `virtual void activemq::commands::ActiveMQStreamMessage::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

*src* - Source Object

Reimplemented from `activemq::commands::Message` (p. 2602).

**6.71.2.5** `virtual bool activemq::commands::ActiveMQStreamMessage::equals ( const DataStructure * value ) const [virtual]`

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 423).

**6.71.2.6** `virtual unsigned char activemq::commands::ActiveMQStreamMessage::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new `DataStructure` (p. 1713) type copy.

Reimplemented from `activemq::commands::Message` (p. 2604).

**6.71.2.7** `virtual void activemq::commands::ActiveMQStreamMessage::onSend ( ) [virtual]`

Store the Data that was written to the stream before a send.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 431).

**6.71.2.8** `virtual bool activemq::commands::ActiveMQStreamMessage::readBoolean ( ) const  
throw ( cms::MessageEOFException, cms::MessageFormatException,  
cms::MessageNotReadableException, cms::CMSException )  
[virtual]`

Reads a Boolean from the Stream message stream.

#### Returns

boolean value from stream

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3763).

**6.71.2.9** `virtual unsigned char activemq::commands::ActiveMQStreamMessage::readByte  
( ) const throw ( cms::MessageEOFException,  
cms::MessageFormatException, cms::MessageNotReadableException,  
cms::CMSException ) [virtual]`

Reads a Byte from the Stream message stream.

#### Returns

unsigned char value from stream

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3764).

**6.71.2.10** `virtual int activemq::commands::ActiveMQStreamMessage::readBytes ( unsigned char * buffer, int length ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [virtual]`

Reads a portion of the Stream message stream.

If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an CMSException is thrown. No bytes will be read from the stream for this exception case.

#### Parameters

<i>buffer</i>	the buffer into which the data is read
<i>length</i>	the number of bytes to read; must be less than or equal to value.length

#### Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3765).

**6.71.2.11** `virtual int activemq::commands::ActiveMQStreamMessage::readBytes ( std::vector< unsigned char > & value ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [virtual]`

Reads a byte array from the Stream message stream.

If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and

so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

### Parameters

<i>value</i>	buffer to place data in
--------------	-------------------------

### Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3764).

```
6.71.2.12 virtual char activemq::commands::ActiveMQStreamMessage::readChar ( ) const
throw ( cms::MessageEOFException, cms::MessageFormatException,
cms::MessageNotReadableException, cms::CMSException )
[virtual]
```

Reads a Char from the Stream message stream.

### Returns

char value from stream

### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3766).

**6.71.2.13** `virtual double activemq::commands::ActiveMQStreamMessage::readDouble ( ) const  
throw ( cms::MessageEOFException, cms::MessageFormatException,  
cms::MessageNotReadableException, cms::CMSException )  
[virtual]`

Reads a 64 bit double from the Stream message stream.

#### Returns

double value from stream

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3766).

**6.71.2.14** `virtual float activemq::commands::ActiveMQStreamMessage::readFloat ( ) const  
throw ( cms::MessageEOFException, cms::MessageFormatException,  
cms::MessageNotReadableException, cms::CMSException )  
[virtual]`

Reads a 32 bit float from the Stream message stream.

#### Returns

double value from stream

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3767).



**6.71.2.15** `virtual int activemq::commands::ActiveMQStreamMessage::readInt ( ) const  
throw ( cms::MessageEOFException, cms::MessageFormatException,  
cms::MessageNotReadableException, cms::CMSException )  
[virtual]`

Reads a 32 bit signed integer from the Stream message stream.

#### Returns

int value from stream

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3767).

**6.71.2.16** `virtual long long activemq::commands::ActiveMQStreamMessage::readLong ( ) const  
throw ( cms::MessageEOFException, cms::MessageFormatException,  
cms::MessageNotReadableException, cms::CMSException )  
[virtual]`

Reads a 64 bit long from the Stream message stream.

#### Returns

long long value from stream

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3768).

**6.71.2.17** `virtual short activemq::commands::ActiveMQStreamMessage::readShort ( ) const  
throw ( cms::MessageEOFException, cms::MessageFormatException,  
cms::MessageNotReadableException, cms::CMSException )  
[virtual]`

Reads a 16 bit signed short from the Stream message stream.

#### Returns

short value from stream

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3769).

**6.71.2.18** `virtual std::string activemq::commands::ActiveMQStreamMessage::readString  
( ) const throw ( cms::MessageEOFException,  
cms::MessageFormatException, cms::MessageNotReadableException,  
cms::CMSException ) [virtual]`

Reads an ASCII String from the Stream message stream.

#### Returns

String from stream

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3769).

**6.71.2.19** `virtual unsigned short activemq::commands::ActiveMQStreamMessage::readUnsignedShort ( ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [virtual]`

Reads a 16 bit unsigned short from the Stream message stream.

#### Returns

unsigned short value from stream

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3770).

**6.71.2.20** `virtual void activemq::commands::ActiveMQStreamMessage::reset ( ) throw ( cms::CMSException ) [virtual]`

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

#### Exceptions

<i>CMSException</i>	
---------------------	--

**6.71.2.21** `virtual std::string activemq::commands::ActiveMQStreamMessage::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 2611).

**6.71.2.22** `virtual void activemq::commands::ActiveMQStreamMessage::writeBoolean ( bool value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Writes a boolean to the Stream message stream as a 1-byte value.

The value true is written as the value (byte)1; the value false is written as the value (byte)0.

#### Parameters

<i>value</i>	boolean to write to the stream
--------------	--------------------------------

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3770).

**6.71.2.23** `virtual void activemq::commands::ActiveMQStreamMessage::writeByte ( unsigned char value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Writes a byte to the Stream message stream as a 1-byte value.

#### Parameters

<i>value</i>	byte to write to the stream
--------------	-----------------------------

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3771).

**6.71.2.24** `virtual void activemq::commands::ActiveMQStreamMessage::writeBytes ( const unsigned char * value, int offset, int length ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Writes a portion of a byte array to the Stream message stream.

size as the number of bytes to write.

**Parameters**

<i>value</i>	bytes to write to the stream
<i>offset</i>	the initial offset within the byte array
<i>length</i>	the number of bytes to use

**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3771).

**6.71.2.25** `virtual void activemq::commands::ActiveMQStreamMessage::writeBytes ( const std::vector< unsigned char > & value ) throw ( cms::MessageNotWriteableException, cms::CMSException )`  
[virtual]

Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.

**Parameters**

<i>value</i>	bytes to write to the stream
--------------	------------------------------

**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3772).

**6.71.2.26** `virtual void activemq::commands::ActiveMQStreamMessage::writeChar ( char value ) throw ( cms::MessageNotWriteableException, cms::CMSException )`  
[virtual]

Writes a char to the Stream message stream as a 1-byte value.

**Parameters**

<i>value</i>	char to write to the stream
--------------	-----------------------------

**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
---------------------	--

<i>MessageNotWriteableException</i>	- if the message is in read-only mode.
-------------------------------------	--

Implements **cms::StreamMessage** (p. 3772).

**6.71.2.27** `virtual void activemq::commands::ActiveMQStreamMessage::writeDouble ( double value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Writes a double to the Stream message stream as a 8 byte value.

#### Parameters

<i>value</i>	double to write to the stream
--------------	-------------------------------

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3772).

**6.71.2.28** `virtual void activemq::commands::ActiveMQStreamMessage::writeFloat ( float value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Writes a float to the Stream message stream as a 4 byte value.

#### Parameters

<i>value</i>	float to write to the stream
--------------	------------------------------

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3773).

**6.71.2.29** `virtual void activemq::commands::ActiveMQStreamMessage::writeInt ( int value )  
throw ( cms::MessageNotWriteableException, cms::CMSException )  
[virtual]`

Writes a signed int to the Stream message stream as a 4 byte value.

#### Parameters

<i>value</i>	signed int to write to the stream
--------------	-----------------------------------

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3773).

**6.71.2.30** `virtual void activemq::commands::ActiveMQStreamMessage::writeLong ( long long value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Writes a long long to the Stream message stream as a 8 byte value.

#### Parameters

<i>value</i>	signed long long to write to the stream
--------------	---

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3774).

**6.71.2.31** `virtual void activemq::commands::ActiveMQStreamMessage::writeShort ( short value ) throw ( cms::MessageNotWriteableException, cms::CMSException )  
[virtual]`

Writes a signed short to the Stream message stream as a 2 byte value.

#### Parameters

<i>value</i>	signed short to write to the stream
--------------	-------------------------------------

**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3774).

**6.71.2.32** `virtual void activemq::commands::ActiveMQStreamMessage::writeString ( const std::string & value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Writes an ASCII String to the Stream message stream.

**Parameters**

<i>value</i>	String to write to the stream
--------------	-------------------------------

**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3774).

**6.71.2.33** `virtual void activemq::commands::ActiveMQStreamMessage::writeUnsignedShort ( unsigned short value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Writes a unsigned short to the Stream message stream as a 2 byte value.

**Parameters**

<i>value</i>	unsigned short to write to the stream
--------------	---------------------------------------

**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3775).



6.71.3.1 `const unsigned char activemq::commands::ActiveMQStreamMessage::ID_-  
 ACTIVEMQSTREAMMESSAGE = 27` `[static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQStreamMessage.h`

## 6.72 activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller**  
 (p. 553).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQStreamMessageMar
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller`:

### Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.72.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p.553). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.72.2 Constructor & Destructor Documentation

- 6.72.2.1 **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller**  
 ( ) [inline]
- 6.72.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller**  
 ( ) [inline, virtual]

### 6.72.3 Member Function Documentation

- 6.72.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

- 6.72.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

## 6.72 ac-

**activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller**

### Class Reference

555

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.72.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2786).

**6.72.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2786).

**6.72.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2787).

**6.72.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2788).

## 6.73 ac-

**activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller**

### Class Reference

557

6.72.3.7 **virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::tightUnmarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \*  
*bs* ) **throw ( decaf::io::IOException )** [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller**  
(p. 2788).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQStreamMessageMarshaller.h

## 6.73 **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller**  
(p. 557).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQStreamMessageMar
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller**:

### Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.73.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 557). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.73.2 Constructor & Destructor Documentation

- 6.73.2.1 **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller**  
( ) [inline]
- 6.73.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller**  
( ) [inline, virtual]

### 6.73.3 Member Function Documentation

- 6.73.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

**6.73 activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller**  
**Class Reference** **559**  
**Returns**

---

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.73.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.73.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2799).

**6.73.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2800).

```
6.73.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2800).

```
6.73.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink



## 6.74 activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller Class Reference 561

<i>bs</i>	- BooleanStream stream used to pack bits from the wire.
-----------	---

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2801).

**6.73.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2802).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQStreamMessageMarshaller.h

## 6.74 activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 561).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQStreamMessageMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller:

## Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.74.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p.561). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.74.2.1 **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller**  
( ) [inline]

6.74.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller**  
( ) [inline, virtual]

### 6.74.3 Member Function Documentation

6.74.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.74.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.74.3.3 **virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2795).

**6.74.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2795).

**6.74.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.74 ac-

**activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller**

### Class Reference

565

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2796).

**6.74.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2797).

**6.74.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2797).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQStreamMessageMarshaller.h

## 6.75 activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 566).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQStreamM
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller**:

### Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 566). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.75.2 Constructor & Destructor Documentation

6.75.2.1 **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller**  
 ( ) [inline]

6.75.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller**  
 ( ) [inline, virtual]

## 6.75.3 Member Function Documentation

6.75.3.1 **virtual commands::DataStructure\* ac-  
 tivemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.75.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.75.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::looseMarshal**  
 ( **OpenWireFormat \* wireFormat**, **commands::DataStructure**  
 \* **dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2781).

**6.75.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2782).

**6.75.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2783).

**6.75.3.6** virtual void **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::tightMarshal2** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2783).

**6.75.3.7** virtual void **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2784).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQStreamMessageMarshaller.h`

## 6.76 **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 570).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQStreamMessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller**:

**Public Member Functions**

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.76.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 570). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.76.2 Constructor & Destructor Documentation

6.76.2.1 **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller**  
( ) [inline]

6.76.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller**  
( ) [inline, virtual]

### 6.76.3 Member Function Documentation

6.76.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.76.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.76.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2790).

**6.76.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2791).

## 6.76 ac-

**activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller**

### Class Reference

573

**6.76.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2791).

**6.76.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2792).

```
6.76.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2793).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQStreamMessageMarshaller.h

## 6.77 activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 574).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQStreamM
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller**:

#### Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.77.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 574). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.77.2 Constructor & Destructor Documentation

6.77.2.1 **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller**  
 ( ) [inline]

6.77.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller**  
 ( ) [inline, virtual]

### 6.77.3 Member Function Documentation

6.77.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.77.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.77.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2804).

**6.77.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---



## 6.77 activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller

### Class Reference 577

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2804).

**6.77.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2805).

**6.77.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink

<i>bs</i>	- BooleanStream stream used to pack bits from the wire.
-----------	---

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2806).

```
6.77.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2806).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQStreamMessageMarshaller.h

## 6.78 activemq::commands::ActiveMQTempDestination Class Reference

```
#include <src/main/activemq/commands/ActiveMQTempDestination.h>
```

Inheritance diagram for **activemq::commands::ActiveMQTempDestination**:

### Public Member Functions

- **ActiveMQTempDestination** ()

- **ActiveMQTempDestination** (const std::string &name)
- virtual ~**ActiveMQTempDestination** ()
- virtual unsigned char **getDataStructureType** () const

*Get the **DataSet** (p. 1713) Type as defined in CommandTypes.h.*

- virtual **ActiveMQTempDestination** \* **cloneDataSet** () const

*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*

- virtual void **copyDataSet** (const **DataSet** \*src)

*Copy the contents of the passed object into this objects members, overwriting any existing data.*

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataSet** (p. 1713) such as its type and value of its elements.*

- virtual bool **equals** (const **DataSet** \*value) const

*Compares the **DataSet** (p. 1713) passed in to this one, and returns if they are equivalent.*

- virtual void **close** () throw ( cms::CMSException )

*Closes down this Destination resulting in a call to dispose of the TempDestination resource at the Broker.*

- void **setConnection** (core::ActiveMQConnection \*connection)

*Sets the Parent Connection that is notified when this destination is destroyed.*

## Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQTEMPDESTINATION** = 0

## Protected Attributes

- core::ActiveMQConnection \* **connection**

*Connection that we call back on close to allow this resource to be cleaned up correctly at this end and at the Broker End.*

## 6.78.1 Constructor & Destructor Documentation

6.78.1.1 `activemq::commands::ActiveMQTempDestination::ActiveMQTempDestination ( )`

6.78.1.2 `activemq::commands::ActiveMQTempDestination::ActiveMQTempDestination ( const std::string & name )`

6.78.1.3 `virtual activemq::commands::ActiveMQTempDestination::~~ActiveMQTempDestination ( ) [virtual]`

## 6.78.2 Member Function Documentation

6.78.2.1 `virtual ActiveMQTempDestination* activemq::commands::ActiveMQTempDestination::cloneDataStructure ( ) const [inline, virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Reimplemented from `activemq::commands::ActiveMQDestination` (p. 315).

Reimplemented in `activemq::commands::ActiveMQTempQueue` (p. 608), and `activemq::commands::ActiveMQTempTopic` (p. 638).

6.78.2.2 `virtual void activemq::commands::ActiveMQTempDestination::close ( ) throw ( cms::CMSException ) [virtual]`

Closes down this Destination resulting in a call to dispose of the TempDestination resource at the Broker.

This should only be called when the user is certain that they are finished with this destination. The TempDestination is not closed automatically on shutdown. throws `cms::CMSException` (p. 1190)

Implements `cms::Closeable` (p. 1180).

6.78.2.3 `virtual void activemq::commands::ActiveMQTempDestination::copyDataStructure ( const DataStructure * src ) [inline, virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

### Returns

src - Source Object

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 316).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 609), and **activemq::commands::ActiveMQTempTopic** (p. 639).

References **activemq::commands::ActiveMQDestination::copyDataStructure()**.

**6.78.2.4** `virtual bool activemq::commands::ActiveMQTempDestination::equals ( const DataStructure * value ) const [inline, virtual]`

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 317).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 609), and **activemq::commands::ActiveMQTempTopic** (p. 639).

References **activemq::commands::ActiveMQDestination::equals()**.

**6.78.2.5** `virtual unsigned char activemq::commands::ActiveMQTempDestination::getDataStructureType ( ) const [virtual]`

Get the **DataStructure** (p. 1713) Type as defined in CommandTypes.h.

#### Returns

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 317).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 610), and **activemq::commands::ActiveMQTempTopic** (p. 640).

**6.78.2.6** `void activemq::commands::ActiveMQTempDestination::setConnection ( core::ActiveMQConnection * connection ) [inline]`

Sets the Parent Connection that is notified when this destination is destroyed.

#### Parameters

<i>connection</i>	- The parent connection.
-------------------	--------------------------

**6.78.2.7** `virtual std::string activemq::commands::ActiveMQTempDestination::toString ( ) const`  
`[virtual]`

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 321).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 611), and **activemq::commands::ActiveMQTempTopic** (p. 641).

### 6.78.3 Field Documentation

**6.78.3.1** `core::ActiveMQConnection* activemq::commands::ActiveMQTempDestination::connection`  
`[protected]`

Connection that we call back on close to allow this resource to be cleaned up correctly at this end and at the Broker End.

**6.78.3.2** `const unsigned char activemq::commands::ActiveMQTempDestination::ID -`  
`ACTIVEMQTEMPDESTINATION = 0` `[static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTempDestination.h`

## 6.79 **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 582).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDes
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller**:

#### Public Member Functions

- **ActiveMQTempDestinationMarshaller** ( )

- 
- virtual `~ActiveMQTempDestinationMarshaller()`
  - virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.79.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 582). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.79.2 Constructor & Destructor Documentation

6.79.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller ( ) [inline]`

6.79.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::~~ActiveMQTempDestinationMarshaller ( ) [inline, virtual]`

## 6.79.3 Member Function Documentation

6.79.3.1 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 325).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 613), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 643).

6.79.3.2 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source



## 6.79 activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller

### Class Reference 585

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 325).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 613), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 643).

**6.79.3.3** `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 326).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 614), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 644).

**6.79.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 326).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 614), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 644).

```
6.79.3.5 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 327).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 615), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 645).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h

6.80 ac-

ativemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller

Class Reference

6.80 activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller<sup>587</sup>

## Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 587).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller:

### Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshall an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.80.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 587). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.80.2 Constructor & Destructor Documentation

6.80.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller ( ) [inline]`

6.80.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::~~ActiveMQTempDestinationMarshaller ( ) [inline, virtual]`

## 6.80.3 Member Function Documentation

6.80.3.1 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 329).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 617), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 651).

6.80.3.2 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 329).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 617), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 652).

**6.80.3.3** `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 330).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 618), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 652).

**6.80.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 330).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 618), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 653).

```
6.80.3.5 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 331).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 619), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 653).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h

6.81 ac-

ativemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller

Class Reference

6.81 activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller<sup>591</sup>

## Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 591).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationM
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller:

### Public Member Functions

- **ActiveMQTempDestinationMarshaller ()**
- virtual **~ActiveMQTempDestinationMarshaller ()**
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)**  
throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn)** throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut)** throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.81.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 591). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.81.2 Constructor & Destructor Documentation

6.81.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller ( ) [inline]`

6.81.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::~~ActiveMQTempDestinationMarshaller ( ) [inline, virtual]`

## 6.81.3 Member Function Documentation

6.81.3.1 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 333).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 621), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 647).

6.81.3.2 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source



## 6.81 activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller

### Class Reference 593

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 333).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 622), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 647).

**6.81.3.3** `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 334).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 622), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 648).

**6.81.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 334).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 623), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 648).

**6.81.3.5** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 335).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 623), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 649).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h`

6.82 ac-

ativemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller

Class Reference

6.82 activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller<sup>595</sup>

## Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 595).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller:

### Public Member Functions

- **ActiveMQTempDestinationMarshaller ()**
- virtual **~ActiveMQTempDestinationMarshaller ()**
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)**  
throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn)** throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut)** throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.82.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 595). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.82.2 Constructor & Destructor Documentation

6.82.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller ( ) [inline]`

6.82.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::~~ActiveMQTempDestinationMarshaller ( ) [inline, virtual]`

## 6.82.3 Member Function Documentation

6.82.3.1 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 337).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 625), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 655).

6.82.3.2 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## 6.82 activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller

### Class Reference 597

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 337).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 626), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 656).

**6.82.3.3** `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 338).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 626), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 656).

**6.82.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 338).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 627), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 657).

```
6.82.3.5 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 339).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 627), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 657).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h

6.83 ac-

ativemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller

Class Reference

6.83 activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller<sup>599</sup>

## Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 599).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller:

### Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshall an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.83.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 599). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.83.2 Constructor & Destructor Documentation

6.83.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller ( ) [inline]`

6.83.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::~~ActiveMQTempDestinationMarshaller ( ) [inline, virtual]`

### 6.83.3 Member Function Documentation

6.83.3.1 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 341).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 630), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 660).

6.83.3.2 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source



## 6.83 activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller

### Class Reference 601

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 341).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 630), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 660).

**6.83.3.3** `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 342).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 631), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 661).

**6.83.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 342).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 631), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 661).

```
6.83.3.5 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 343).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 632), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 662).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h`

6.84 ac-

ativemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller

Class Reference

6.84 activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller<sup>603</sup>

## Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 603).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationM
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller:

### Public Member Functions

- **ActiveMQTempDestinationMarshaller ()**
- virtual **~ActiveMQTempDestinationMarshaller ()**
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)**  
throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn)** throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut)** throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.84.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 603). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.84.2 Constructor & Destructor Documentation

6.84.2.1 `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller ( ) [inline]`

6.84.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::~~ActiveMQTempDestinationMarshaller ( ) [inline, virtual]`

## 6.84.3 Member Function Documentation

6.84.3.1 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 345).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller` (p. 634), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller` (p. 664).

6.84.3.2 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## 6.84 activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller

### Class Reference 605

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 345).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 634), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 664).

**6.84.3.3** `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 346).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 635), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 665).

**6.84.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 346).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 635), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 665).

**6.84.3.5** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 347).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 636), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 666).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationMarshaller.h`

**6.85 activemq::commands::ActiveMQTempQueue Class Reference**

```
#include <src/main/activemq/commands/ActiveMQTempQueue.h>
```

Inheritance diagram for activemq::commands::ActiveMQTempQueue:

## Public Member Functions

- **ActiveMQTempQueue** ()
- **ActiveMQTempQueue** (const std::string &name)
- virtual ~**ActiveMQTempQueue** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the **DataStructure** (p. 1713) Type as defined in CommandTypes.h.*
- virtual **ActiveMQTempQueue** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Converts the Destination Name into a String.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const cms::Destination \* **getCMSDestination** () const
- virtual cms::Destination::DestinationType **getDestinationType** () const  
*Retrieve the Destination Type for this Destination.*
- virtual cms::Destination \* **clone** () const  
*Creates a new instance of this destination type that is a copy of this one, and returns it.*
- virtual void **copy** (const cms::Destination &source)  
*Copies the contents of the given Destination object to this one.*
- virtual const cms::CMSProperties & **getCMSProperties** () const  
*Retrieve any properties that might be part of the destination that was specified.*
- virtual std::string **getQueueName** () const throw ( cms::CMSException )  
*Gets the name of this queue.*
- virtual void **destroy** () throw ( cms::CMSException )  
*Destroy's the Temp Destination at the Broker.*

## Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQTEMPQUEUE** = 102

## 6.85.1 Constructor & Destructor Documentation

6.85.1.1 `activemq::commands::ActiveMQTempQueue::ActiveMQTempQueue ( )`

6.85.1.2 `activemq::commands::ActiveMQTempQueue::ActiveMQTempQueue ( const std::string & name )`

6.85.1.3 `virtual activemq::commands::ActiveMQTempQueue::~~ActiveMQTempQueue ( )`  
[virtual]

## 6.85.2 Member Function Documentation

6.85.2.1 `virtual cms::Destination* activemq::commands::ActiveMQTempQueue::clone ( )`  
`const` [inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

### Returns

cloned copy of this object

Implements **cms::Destination** (p. 1778).

6.85.2.2 `virtual ActiveMQTempQueue* activemq::commands::ActiveMQTempQueue::cloneDataStructure ( )`  
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 580).

6.85.2.3 `virtual void activemq::commands::ActiveMQTempQueue::copy ( const cms::Destination & source )` [inline, virtual]

Copies the contents of the given Destination object to this one.

## Parameters

<i>source</i>	The source Destination object.
---------------	--------------------------------



Implements **cms::Destination** (p. 1778).

**6.85.2.4** `virtual void activemq::commands::ActiveMQTempQueue::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 580).

**6.85.2.5** `virtual void activemq::commands::ActiveMQTempQueue::destroy ( ) throw ( cms::CMSException ) [virtual]`

Destroy's the Temp Destination at the Broker.

#### Exceptions

<i>CMSException</i>	
---------------------	--

Implements **cms::TemporaryQueue** (p. 3872).

**6.85.2.6** `virtual bool activemq::commands::ActiveMQTempQueue::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 581).

**6.85.2.7** `virtual const cms::Destination* activemq::commands::ActiveMQTempQueue::getCMSDestination ( ) const [inline, virtual]`

#### Returns

the **cms::Destination** (p. 1776) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 317).

**6.85.2.8** `virtual const cms::CMSProperties& activemq::commands::ActiveMQTempQueue::getCMSProperties ( ) const [inline, virtual]`

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

#### Returns

const reference to a properties object.

Implements **cms::Destination** (p. 1778).

**6.85.2.9** `virtual unsigned char activemq::commands::ActiveMQTempQueue::getDataStructureType ( ) const [virtual]`

Get the **DataStructure** (p. 1713) Type as defined in CommandTypes.h.

#### Returns

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 581).

**6.85.2.10** `virtual cms::Destination::DestinationType activemq::commands::ActiveMQTempQueue::getDestinationType ( ) const [inline, virtual]`

Retrieve the Destination Type for this Destination.

#### Returns

The Destination Type

Implements **activemq::commands::ActiveMQDestination** (p. 318).

References cms::Destination::TEMPORARY\_QUEUE.

**6.85.2.11** `virtual std::string activemq::commands::ActiveMQTempQueue::getQueueName ( ) const throw ( cms::CMSException ) [inline, virtual]`

Gets the name of this queue.

#### Returns

The queue name.

Implements **cms::TemporaryQueue** (p. 3872).

**6.86 ac-**  
**ativemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller**  
**Class Reference** 611  
**6.85.2.12** `virtual std::string activemq::commands::ActiveMQTempQueue::toString ( ) const`  
`[virtual]`

Converts the Destination Name into a String.

#### Returns

string name

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 582).

### 6.85.3 Field Documentation

**6.85.3.1** `const unsigned char activemq::commands::ActiveMQTempQueue::ID_ -`  
`ACTIVEMQTEMPQUEUE = 102` `[static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTempQueue.h`

## 6.86 activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 611).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshal
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller**:

### Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- **virtual ~ActiveMQTempQueueMarshaller** ()
- **virtual commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- **virtual unsigned char getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- **virtual void tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) **throw** ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.86.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 611).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.86.2 Constructor & Destructor Documentation

6.86.2.1 **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller**  
 ( ) [inline]

6.86.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller**  
 ( ) [inline, virtual]

### 6.86.3 Member Function Documentation

6.86.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.86 ac-

ativemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller

Class Reference

613

6.86.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::getDataStructureType  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaller.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.86.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::looseMarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure  
\* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw (  
decaf::io::IOException) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller**  
(p. 584).

6.86.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::looseUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure  
\* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw (  
decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 584).

```
6.86.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 585).

```
6.86.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 585).

6.87 ac-

ativemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller

Class Reference

615

6.86.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::tightUnmarshal  
( OpenWireFormat \* wireFormat, commands::DataStructure \*  
dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \*  
bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

wireFormat	- describes the wire format of the broker.
dataStructure	- Object to be un-marshaled.
dataIn	- BinaryReader that provides that data.
bs	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

IOException	if an error occurs during the unmarshal.
-------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 586).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshaller.h

## 6.87 activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 615).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshal
```

Inheritance diagram for **ativemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller**:

#### Public Member Functions

- **ActiveMQTempQueueMarshaller ()**
- virtual **~ActiveMQTempQueueMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.87.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 615).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.87.2 Constructor & Destructor Documentation

6.87.2.1 **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller**  
( ) [inline]

6.87.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller**  
( ) [inline, virtual]

### 6.87.3 Member Function Documentation

6.87.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.



**6.87 activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller**  
**Class Reference** **617**  
**Returns**

---

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.87.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::getDataStructureType ( ) const` [virtual]

Get the Data Structure Type that identifies this Marshaller.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.87.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 588).

**6.87.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 588).

```
6.87.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 589).

```
6.87.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink

## 6.88 activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller

### Class Reference

619

<i>bs</i>	- BooleanStream stream used to pack bits from the wire.
-----------	---

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 589).

**6.87.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 590).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h

## 6.88 activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 619).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempQueueMarshal
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller:

## Public Member Functions

- **ActiveMQTempQueueMarshaller ()**
- virtual **~ActiveMQTempQueueMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn)** throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.88.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 619).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

6.88.2.1 **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller**  
( ) [inline]

6.88.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller**  
( ) [inline, virtual]

### 6.88.3 Member Function Documentation

6.88.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.88.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.88.3.3 **virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 592).

```
6.88.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 592).

```
6.88.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.88 ac-

**activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller**

### Class Reference

623

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 593).

**6.88.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 593).

**6.88.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 594).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempQueueMarshaller.h`

## 6.89 activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 624).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller`:

### Public Member Functions

- **ActiveMQTempQueueMarshaller ()**
- virtual **~ActiveMQTempQueueMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn)** throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*



Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 624).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

## 6.89.2 Constructor & Destructor Documentation

**6.89.2.1** `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller  
 ( ) [inline]`

**6.89.2.2** `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller  
 ( ) [inline, virtual]`

## 6.89.3 Member Function Documentation

**6.89.3.1** `virtual commands::DataStructure* ac-  
 tivemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::createObject  
 ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.89.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::getDataStructureType  
 ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.89.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::looseMarshal  
 ( OpenWireFormat * wireFormat, commands::DataStructure  
 * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 596).

```
6.89.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 596).

```
6.89.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**6.89 ac-**  
**ativemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller**  
**Class Reference** **627**  
**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 597).

**6.89.3.6** virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::tightMarshal2  
( OpenWireFormat \* *wireFormat*, commands::DataStructure  
\* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 597).

**6.89.3.7** virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::tightUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \*  
*bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 598).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarshaller.h`

## 6.90 `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQTempQueueMarshaller` (p. 628).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller`:

**Public Member Functions**

- `ActiveMQTempQueueMarshaller ()`
- `virtual ~ActiveMQTempQueueMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaller.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

## 6.90 ac-

**ativemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller**

### Class Reference

629

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

## 6.90.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 628).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.90.2 Constructor & Destructor Documentation

6.90.2.1 **ativemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller**  
( ) [inline]

6.90.2.2 **virtual ativemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller**  
( ) [inline, virtual]

## 6.90.3 Member Function Documentation

6.90.3.1 **virtual commands::DataStructure\* ac-**  
**ativemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **ativemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.90.3.2 **virtual unsigned char ativemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

```
6.90.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 600).

```
6.90.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 600).

6.90 ac-

activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller

Class Reference

631

```
6.90.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 601).

```
6.90.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 601).

```
6.90.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 602).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h`

## 6.91 `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQTempQueueMarshaller` (p. 632).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempQueueMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller`:

#### Public Member Functions

- `ActiveMQTempQueueMarshaller ()`
- `virtual ~ActiveMQTempQueueMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaller.*



## 6.91 ac-

**ativemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller**

### Class Reference

633

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.91.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 632).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.91.2 Constructor & Destructor Documentation

6.91.2.1 **ativemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller**  
( ) [inline]

6.91.2.2 **virtual ativemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller**  
( ) [inline, virtual]

### 6.91.3 Member Function Documentation

6.91.3.1 **virtual commands::DataStructure\* ativemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.91.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.91.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 604).

**6.91.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

## 6.91 ac-

### activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller Class Reference 635

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 604).

```
6.91.3.5 virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::tightMarshal1  
( OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )  
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 605).

```
6.91.3.6 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::tightMarshal2  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataOutputStream * dataOut,  
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink

<i>bs</i>	- BooleanStream stream used to pack bits from the wire.
-----------	---

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 605).

```
6.91.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 606).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempQueueMarshaller.h`

## 6.92 activemq::commands::ActiveMQTempTopic Class Reference

```
#include <src/main/activemq/commands/ActiveMQTempTopic.h>
```

Inheritance diagram for `activemq::commands::ActiveMQTempTopic`:

### Public Member Functions

- `ActiveMQTempTopic ()`
- `ActiveMQTempTopic (const std::string &name)`

- virtual `~ActiveMQTempTopic ()`
- virtual unsigned char `getDataStructureType ()` const  
*Get the **DataSetructure** (p. 1713) Type as defined in CommandTypes.h.*
- virtual `ActiveMQTempTopic * cloneDataSetructure ()` const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void `copyDataSetructure (const DataSetructure *src)`  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string `toString ()` const  
*Converts the Destination Name into a String.*
- virtual bool `equals (const DataSetructure *value)` const  
*Compares the **DataSetructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const `cms::Destination * getCMSDestination ()` const
- virtual `cms::Destination::DestinationType getDestinationType ()` const  
*Retrieve the Destination Type for this Destination.*
- virtual `cms::Destination * clone ()` const  
*Creates a new instance of this destination type that is a copy of this one, and returns it.*
- virtual void `copy (const cms::Destination &source)`  
*Copies the contents of the given Destinastion object to this one.*
- virtual const `cms::CMSProperties & getCMSProperties ()` const  
*Retrieve any properties that might be part of the destination that was specified.*
- virtual std::string `getTopicName ()` const throw ( cms::CMSException )  
*Gets the name of this topic.*
- virtual void `destroy ()` throw ( cms::CMSException )  
*Destroy's the Temp Destination at the Broker.*

### Static Public Attributes

- static const unsigned char `ID_ACTIVEMQTEMPTOPIC = 103`

## 6.92.1 Constructor & Destructor Documentation

6.92.1.1 `activemq::commands::ActiveMQTempTopic::ActiveMQTempTopic ( )`

6.92.1.2 `activemq::commands::ActiveMQTempTopic::ActiveMQTempTopic ( const std::string & name )`

6.92.1.3 `virtual activemq::commands::ActiveMQTempTopic::~~ActiveMQTempTopic ( )`  
[virtual]

## 6.92.2 Member Function Documentation

6.92.2.1 `virtual cms::Destination* activemq::commands::ActiveMQTempTopic::clone ( )`  
`const` [inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

### Returns

cloned copy of this object

Implements `cms::Destination` (p. 1778).

6.92.2.2 `virtual ActiveMQTempTopic* activemq::commands::ActiveMQTempTopic::cloneDataStructure ( )`  
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Reimplemented from `activemq::commands::ActiveMQTempDestination` (p. 580).

6.92.2.3 `virtual void activemq::commands::ActiveMQTempTopic::copy ( const cms::Destination & source )` [inline, virtual]

Copies the contents of the given Destination object to this one.

### Parameters

<i>source</i>	The source Destination object.
---------------	--------------------------------

Implements `cms::Destination` (p. 1778).

**6.92.2.4** `virtual void activemq::commands::ActiveMQTempTopic::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 580).

**6.92.2.5** `virtual void activemq::commands::ActiveMQTempTopic::destroy ( ) throw ( cms::CMSException ) [virtual]`

Destroy's the Temp Destination at the Broker.

#### Exceptions

<i>CMSException</i>	
---------------------	--

Implements **cms::TemporaryTopic** (p. 3873).

**6.92.2.6** `virtual bool activemq::commands::ActiveMQTempTopic::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 581).

**6.92.2.7** `virtual const cms::Destination* activemq::commands::ActiveMQTempTopic::getCMSDestination ( ) const [inline, virtual]`

#### Returns

the **cms::Destination** (p. 1776) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 317).

**6.92.2.8** `virtual const cms::CMSProperties& activemq::commands::ActiveMQTempTopic::getCMSProperties ( ) const [inline, virtual]`

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

#### Returns

const reference to a properties object.

Implements **cms::Destination** (p. 1778).

**6.92.2.9** `virtual unsigned char activemq::commands::ActiveMQTempTopic::getDataStructureType ( ) const [virtual]`

Get the **DataStructure** (p. 1713) Type as defined in CommandTypes.h.

#### Returns

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 581).

**6.92.2.10** `virtual cms::Destination::DestinationType activemq::commands::ActiveMQTempTopic::getDestinationType ( ) const [inline, virtual]`

Retrieve the Destination Type for this Destination.

#### Returns

The Destination Type

Implements **activemq::commands::ActiveMQDestination** (p. 318).

References cms::Destination::TEMPORARY\_TOPIC.

**6.92.2.11** `virtual std::string activemq::commands::ActiveMQTempTopic::getTopicName ( ) const throw ( cms::CMSException ) [inline, virtual]`

Gets the name of this topic.

#### Returns

The topic name.

Implements **cms::TemporaryTopic** (p. 3874).



## 6.93

**activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller**

### Class Reference

641

6.92.2.12 `virtual std::string activemq::commands::ActiveMQTempTopic::toString ( ) const`  
[virtual]

Converts the Destination Name into a String.

### Returns

string name

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 582).

## 6.92.3 Field Documentation

6.92.3.1 `const unsigned char activemq::commands::ActiveMQTempTopic::ID_-`  
`ACTIVEMQTEMPTOPIC = 103` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTempTopic.h`

## 6.93 activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 641).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshal
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller**:

### Public Member Functions

- **ActiveMQTempTopicMarshaller ()**
- **virtual ~ActiveMQTempTopicMarshaller ()**
- **virtual commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- **virtual unsigned char getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- **virtual void tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.93.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 641).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.93.2 Constructor & Destructor Documentation

6.93.2.1 **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller**  
 ( ) [inline]

6.93.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller**  
 ( ) [inline, virtual]

### 6.93.3 Member Function Documentation

6.93.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

## 6.93

**activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller**

### Class Reference

643

**6.93.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaller.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.93.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 584).

**6.93.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 584).

```
6.93.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 585).

```
6.93.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 585).

## 6.94

**activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller**

### Class Reference

645

6.93.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 586).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshaller.h`

## 6.94 **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 645).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempTopicMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller**:

### Public Member Functions

- **ActiveMQTempTopicMarshaller ()**
- **virtual ~ActiveMQTempTopicMarshaller ()**
- **virtual commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- **virtual unsigned char getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaller.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.94.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 645).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.94.2 Constructor & Destructor Documentation

6.94.2.1 **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller**  
( ) [inline]

6.94.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller**  
( ) [inline, virtual]

### 6.94.3 Member Function Documentation

6.94.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

## 6.94

**activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller**

**Class Reference**

**647**

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.94.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.94.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 592).

**6.94.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 592).

```
6.94.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 593).

```
6.94.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink



## 6.95

### activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller Class Reference 649

<i>bs</i>	- BooleanStream stream used to pack bits from the wire.
-----------	---

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 593).

**6.94.3.7** virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 594).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempTopicMarshaller.h

## 6.95 activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 649).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller:

## Public Member Functions

- **ActiveMQTempTopicMarshaller ()**
- virtual **~ActiveMQTempTopicMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn)** throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.95.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 649).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.95

activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller

Class Reference

651

## 6.95.2 Constructor & Destructor Documentation

6.95.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller ( ) [inline]`

6.95.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller ( ) [inline, virtual]`

## 6.95.3 Member Function Documentation

6.95.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.95.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.95.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 588).

```
6.95.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 588).

```
6.95.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.95

**activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller**

### Class Reference

653

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 589).

```
6.95.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataOutputStream * dataOut,
 utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 589).

```
6.95.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
 dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
 bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 590).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h

## 6.96 activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 654).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller`:

### Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

6.96

**activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller**  
Class Reference 655

### 6.96.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 654).  
NOTE!: This file is auto generated - do not modify! if you need to make a change,  
please see the Java Classes in the activemq-openwire-generator module

### 6.96.2 Constructor & Destructor Documentation

6.96.2.1 **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller**  
( ) [inline]

6.96.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller**  
( ) [inline, virtual]

### 6.96.3 Member Function Documentation

6.96.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.96.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.96.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 596).

**6.96.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 596).

**6.96.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



6.96

**activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller****Class Reference****657****Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 597).

**6.96.3.6** virtual void **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::tightMarshal2**  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
 \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*,  
**utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 597).

**6.96.3.7** virtual void **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::tightUnmarshal**  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \*  
*bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 598).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller.h`

## 6.97 `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 658).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller`:

**Public Member Functions**

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

## 6.97

**activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller**

### Class Reference

659

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.97.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 658).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.97.2 Constructor & Destructor Documentation

6.97.2.1 **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller**  
( ) [inline]

6.97.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller**  
( ) [inline, virtual]

### 6.97.3 Member Function Documentation

6.97.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.97.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

```
6.97.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 600).

```
6.97.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 600).

6.97

activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller

Class Reference

661

```
6.97.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::tightMarshal1  
( OpenWireFormat * wireFormat, commands::DataStructure *  
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )  
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 601).

```
6.97.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::tightMarshal2  
( OpenWireFormat * wireFormat, commands::DataStructure  
  * dataStructure, decaf::io::DataOutputStream * dataOut,  
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 601).

```
6.97.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 602).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller.h`

## 6.98 `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQTempTopicMarshaller` (p. 662).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempTopicMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller`:

#### Public Member Functions

- `ActiveMQTempTopicMarshaller ()`
- `virtual ~ActiveMQTempTopicMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaller.*

## 6.98

**activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller**

### Class Reference

663

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.98.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 662).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.98.2 Constructor & Destructor Documentation

6.98.2.1 **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller**  
( ) [inline]

6.98.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller**  
( ) [inline, virtual]

### 6.98.3 Member Function Documentation

6.98.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.98.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.98.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 604).

**6.98.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---



## 6.98

### activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller Class Reference 665

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 604).

```
6.98.3.5  virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::tightMarshal1  
( OpenWireFormat * wireFormat, commands::DataStructure *  
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )  
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 605).

```
6.98.3.6  virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::tightMarshal2  
( OpenWireFormat * wireFormat, commands::DataStructure  
  * dataStructure, decaf::io::DataOutputStream * dataOut,  
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink

<i>bs</i>	- BooleanStream stream used to pack bits from the wire.
-----------	---

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 605).

```
6.98.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 606).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempTopicMarshaller.h`

**6.99 activemq::commands::ActiveMQTextMessage Class Reference**

```
#include <src/main/activemq/commands/ActiveMQTextMessage.h>
```

Inheritance diagram for `activemq::commands::ActiveMQTextMessage`:

**Public Member Functions**

- `ActiveMQTextMessage ()`
- `virtual ~ActiveMQTextMessage ()`

- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ActiveMQTextMessage** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual void **clearBody** () throw ( cms::CMSException )  
*Clears out the body of the message.*
- virtual void **beforeMarshal** (wireformat::WireFormat \*wireFormat) throw ( decaf::io::IOException )  
*Performs any cleanup or other tasks that must be done before the **Message** (p. 2596) is marshalled to its binary WireFormat version.*
- virtual unsigned int **getSize** () const  
*Returns the Size of this message in Bytes.*
- virtual cms::TextMessage \* **clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*
- virtual std::string **getText** () const throw ( cms::CMSException )  
*Gets the message character buffer.*
- virtual void **setText** (const char \*msg) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Sets the message contents, does not take ownership of the passed char\*, but copies it instead.*
- virtual void **setText** (const std::string &msg) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Sets the message contents.*

## Data Fields

- `std::auto_ptr< std::string > text`

## Static Public Attributes

- static const unsigned char `ID_ACTIVEMQTEXTMESSAGE` = 28

## 6.99.1 Constructor & Destructor Documentation

**6.99.1.1** `activemq::commands::ActiveMQTextMessage::ActiveMQTextMessage ( )`

**6.99.1.2** `virtual activemq::commands::ActiveMQTextMessage::~~ActiveMQTextMessage ( )`  
[virtual]

## 6.99.2 Member Function Documentation

**6.99.2.1** `virtual void activemq::commands::ActiveMQTextMessage::beforeMarshal ( wireformat::WireFormat * wireFormat ) throw ( decaf::io::IOException )`  
[virtual]

Performs any cleanup or other tasks that must be done before the `Message` (p. 2596) is marshalled to its binary `WireFormat` version.

### Parameters

<i>wireFormat</i>	the <code>WireFormat</code> instance that is marshalling this message.
-------------------	--

Implements `activemq::wireformat::MarshalAware` (p. 2565).

**6.99.2.2** `virtual void activemq::commands::ActiveMQTextMessage::clearBody ( ) throw ( cms::CMSException )` [virtual]

Clears out the body of the message.

This does not clear the headers or properties.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 422).

**6.99.2.3** `virtual cms::TextMessage* activemq::commands::ActiveMQTextMessage::clone ( ) const` [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

## Returns

new copy of this message

Implements **cms::Message** (p. 2620).

**6.99.2.4** `virtual ActiveMQTextMessage* activemq::commands::ActiveMQTextMessage::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 2601).

**6.99.2.5** `virtual void activemq::commands::ActiveMQTextMessage::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 2602).

**6.99.2.6** `virtual bool activemq::commands::ActiveMQTextMessage::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 423).

**6.99.2.7** `virtual unsigned char activemq::commands::ActiveMQTextMessage::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns**

new **DataStructure** (p. 1713) type copy.

Reimplemented from **activemq::commands::Message** (p. 2604).

**6.99.2.8** `virtual unsigned int activemq::commands::ActiveMQTextMessage::getSize ( ) const`  
[virtual]

Returns the Size of this message in Bytes.

**Returns**

number of bytes this message equates to.

Reimplemented from **activemq::commands::Message** (p. 2606).

**6.99.2.9** `virtual std::string activemq::commands::ActiveMQTextMessage::getText ( ) const`  
`throw ( cms::CMSEException )` [virtual]

Gets the message character buffer.

**Returns**

The message character buffer.

**Exceptions**

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::TextMessage** (p. 3875).

**6.99.2.10** `virtual void activemq::commands::ActiveMQTextMessage::setText ( const std::string &`  
`msg ) throw ( cms::MessageNotWriteableException, cms::CMSEException`  
`)` [virtual]

Sets the message contents.

**Parameters**

<i>msg</i>	The message buffer.
------------	---------------------

**Exceptions**

<i>CMSEException</i>	- if an internal error occurs.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode..

Implements **cms::TextMessage** (p. 3875).

## 6.100 ac-

### activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller Class Reference 671

6.99.2.11 `virtual void activemq::commands::ActiveMQTextMessage::setText ( const char * msg ) throw ( cms::MessageNotWriteableException, cms::CMSException )`  
[virtual]

Sets the message contents, does not take ownership of the passed char\*, but copies it instead.

#### Parameters

<i>msg</i>	The message buffer.
------------	---------------------

#### Exceptions

<i>CMSException</i>	- if an internal error occurs.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode..

Implements **cms::TextMessage** (p. 3876).

6.99.2.12 `virtual std::string activemq::commands::ActiveMQTextMessage::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 2611).

### 6.99.3 Field Documentation

6.99.3.1 `const unsigned char activemq::commands::ActiveMQTextMessage::ID_-ACTIVEMQTEXTMESSAGE = 28` [static]

6.99.3.2 `std::auto_ptr<std::string> activemq::commands::ActiveMQTextMessage::text`  
[mutable]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/ActiveMQTextMessage.h

## 6.100 activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 671).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller`:

## Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.100.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 671).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module



6.100.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller ( ) [inline]`

6.100.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller ( ) [inline, virtual]`

### 6.100.3 Member Function Documentation

6.100.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.100.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.100.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2786).

**6.100.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2786).

**6.100.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.100 ac-

**activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller**

### Class Reference

675

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2787).

```
6.100.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2788).

```
6.100.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2788).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h`

## 6.101 `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 676).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMes
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`:

### Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.101.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 676).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.101.2 Constructor & Destructor Documentation

**6.101.2.1** `ativemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller  
 ( ) [inline]`

**6.101.2.2** `virtual ativemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller  
 ( ) [inline, virtual]`

### 6.101.3 Member Function Documentation

**6.101.3.1** `virtual commands::DataStructure* ac-  
 tivemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::createObject  
 ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **ativemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.101.3.2** `virtual unsigned char ativemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::getDataStructureType  
 ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **ativemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.101.3.3** `virtual void ativemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::looseMarshal  
 ( OpenWireFormat * wireFormat, commands::DataStructure  
 * dataStructure, decaf::io::DataOutputStream * dataOut ) throw (  
 decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2795).

**6.101.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2795).

**6.101.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## 6.101 activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller

### Class Reference

---

#### Returns

679

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2796).

**6.101.3.6** virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::tightMarshal2 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2797).

**6.101.3.7** virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2797).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTextMessageMarshaller.h**

## 6.102 **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 680).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMes
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller**:

**Public Member Functions**

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*



## 6.102 ac-

**ativemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller**

### Class Reference

681

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

## 6.102.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 680).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.102.2 Constructor & Destructor Documentation

6.102.2.1 **ativemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller**  
( ) [inline]

6.102.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller**  
( ) [inline, virtual]

## 6.102.3 Member Function Documentation

6.102.3.1 **virtual commands::DataStructure\* ac-**  
**ativemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **ativemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.102.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.102.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2799).

**6.102.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2800).

## 6.102 ac-

ativemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller

### Class Reference

683

```
6.102.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::tightMarshal1  
( OpenWireFormat * wireFormat, commands::DataStructure *  
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )  
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2800).

```
6.102.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::tightMarshal2  
( OpenWireFormat * wireFormat, commands::DataStructure  
  * dataStructure, decaf::io::DataOutputStream * dataOut,  
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )  
[virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2801).

```
6.102.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2802).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h

## 6.103 activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 684).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTextMes
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller**:

#### Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

## 6.103 ac-

**ativemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller**

### Class Reference

685

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

## 6.103.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 684).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.103.2 Constructor & Destructor Documentation

6.103.2.1 **ativemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller**  
( ) [inline]

6.103.2.2 **virtual ativemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller**  
( ) [inline, virtual]

## 6.103.3 Member Function Documentation

6.103.3.1 **virtual commands::DataStructure\* ativemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.103.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.103.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2781).

**6.103.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

## 6.103 ac-

### activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller

#### Class Reference

687

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2782).

```
6.103.3.5  virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2783).

```
6.103.3.6  virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2783).

**6.103.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2784).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTextMessageMarshaller.h`

## 6.104 **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 688).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTextMes
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller**:



- **ActiveMQTextMessageMarshaller ()**
- virtual **~ActiveMQTextMessageMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.104.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 688).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.104.2 Constructor & Destructor Documentation

6.104.2.1 `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller ( ) [inline]`

6.104.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller ( ) [inline, virtual]`

### 6.104.3 Member Function Documentation

6.104.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.104.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.104.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## 6.104 ac-

**activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller**  
**Class Reference** **691**  
**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2804).

**6.104.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2804).

**6.104.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2805).

```
6.104.3.6 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2806).

```
6.104.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2806).

The documentation for this class was generated from the following file:

## 6.105 ac-

**ativemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller**

### Class Reference

693

- [src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTextMessageMarshaller.h](#)

## 6.105 **ativemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 693).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarsh
```

Inheritance diagram for **ativemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller**:

### Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.105.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 693).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.105.2 Constructor & Destructor Documentation

6.105.2.1 **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller**  
 ( ) [inline]

6.105.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller**  
 ( ) [inline, virtual]

### 6.105.3 Member Function Documentation

6.105.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.105.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.105.3.3 **virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::looseMarshal**  
 ( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

## 6.105 activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller

### Class Reference 695

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2790).

**6.105.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2791).

**6.105.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2791).

```
6.105.3.6  virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2792).

```
6.105.3.7  virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.



**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2793).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarshaller.h

**6.106 activemq::commands::ActiveMQTopic Class Reference**

```
#include <src/main/activemq/commands/ActiveMQTopic.h>
```

Inheritance diagram for activemq::commands::ActiveMQTopic:

**Public Member Functions**

- **ActiveMQTopic** ()
- **ActiveMQTopic** (const std::string &name)
- virtual ~**ActiveMQTopic** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the **DataStructure** (p. 1713) Type as defined in CommandTypes.h.*
- virtual **ActiveMQTopic** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const **cms::Destination** \* **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const  
*Retrieve the Destination Type for this Destination.*

- virtual **cms::Destination \* clone ()** const  
*Creates a new instance of this destination type that is a copy of this one, and returns it.*
- virtual void **copy** (const **cms::Destination** &source)  
*Copies the contents of the given Destination object to this one.*
- virtual const **cms::CMSProperties & getCMSProperties ()** const  
*Retrieve any properties that might be part of the destination that was specified.*
- virtual std::string **getTopicName ()** const throw ( cms::CMSException )  
*Gets the name of this topic.*

### Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQTOPIC** = 101

### 6.106.1 Constructor & Destructor Documentation

- 6.106.1.1 **activemq::commands::ActiveMQTopic::ActiveMQTopic ( )**
- 6.106.1.2 **activemq::commands::ActiveMQTopic::ActiveMQTopic ( const std::string & name )**
- 6.106.1.3 **virtual activemq::commands::ActiveMQTopic::~~ActiveMQTopic ( )** [virtual]

### 6.106.2 Member Function Documentation

- 6.106.2.1 **virtual cms::Destination\* activemq::commands::ActiveMQTopic::clone ( )** const  
[inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

#### Returns

cloned copy of this object

Implements **cms::Destination** (p. 1778).

- 6.106.2.2 **virtual ActiveMQTopic\* activemq::commands::ActiveMQTopic::cloneDataStructure ( )** const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 315).

**6.106.2.3** `virtual void activemq::commands::ActiveMQTopic::copy ( const cms::Destination & source ) [inline, virtual]`

Copies the contents of the given Destination object to this one.

#### Parameters

<i>source</i>	The source Destination object.
---------------	--------------------------------

Implements **cms::Destination** (p. 1778).

**6.106.2.4** `virtual void activemq::commands::ActiveMQTopic::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 316).

**6.106.2.5** `virtual bool activemq::commands::ActiveMQTopic::equals ( const DataStructure * value ) const [inline, virtual]`

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 317).

References **activemq::commands::ActiveMQDestination::equals()**.

**6.106.2.6** `virtual const cms::Destination* activemq::commands::ActiveMQTopic::getCMSDestination ( ) const [inline, virtual]`

#### Returns

the **cms::Destination** (p. 1776) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 317).

**6.106.2.7** `virtual const cms::CMSProperties& activemq::commands::ActiveMQTopic::getCMSProperties ( ) const [inline, virtual]`

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

#### Returns

const reference to a properties object.

Implements **cms::Destination** (p. 1778).

**6.106.2.8** `virtual unsigned char activemq::commands::ActiveMQTopic::getDataStructureType ( ) const [virtual]`

Get the **DataStructure** (p. 1713) Type as defined in CommandTypes.h.

#### Returns

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 317).

**6.106.2.9** `virtual cms::Destination::DestinationType activemq::commands::ActiveMQTopic::getDestinationType ( ) const [inline, virtual]`

Retrieve the Destination Type for this Destination.

#### Returns

The Destination Type

Implements **activemq::commands::ActiveMQDestination** (p. 318).

References cms::Destination::TOPIC.

**6.106.2.10** `virtual std::string activemq::commands::ActiveMQTopic::getTopicName ( ) const throw ( cms::CMSExcption ) [inline, virtual]`

Gets the name of this topic.

#### Returns

The topic name.

Implements **cms::Topic** (p. 3931).

## 6.107 activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller Class Reference 701

---

6.106.2.11 `virtual std::string activemq::commands::ActiveMQTopic::toString ( ) const`  
`[virtual]`

Returns a string containing the information for this **DataSet** (p. 1713) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 321).

### 6.106.3 Field Documentation

6.106.3.1 `const unsigned char activemq::commands::ActiveMQTopic::ID_ -`  
`ACTIVEMQTOPIC = 101 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTopic.h`

## 6.107 activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 701).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller**:

### Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataSet** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataSetType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataSet** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.107.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 701). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.107.2 Constructor & Destructor Documentation

6.107.2.1 **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller** ( ) [**inline**]

6.107.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller** ( ) [**inline**, **virtual**]

### 6.107.3 Member Function Documentation

6.107.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::createObject** ( ) **const** [**virtual**]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

## 6.107 activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller

### Class Reference 703

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.107.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::getDataStructureType ( ) const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.107.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 325).

**6.107.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 325).

```
6.107.3.5  virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 326).

```
6.107.3.6  virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



## 6.108 activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller

### Class Reference

705

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 326).

**6.107.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 327).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h`

## 6.108 activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 705).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller**:

#### Public Member Functions

- **ActiveMQTopicMarshaller ()**

- virtual `~ActiveMQTopicMarshaller ()`
- virtual `commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.108.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 705). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.108.2 Constructor & Destructor Documentation

6.108.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller ( ) [inline]`

6.108.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller ( ) [inline, virtual]`

## 6.108.3 Member Function Documentation

6.108.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.108.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.108.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 333).

**6.108.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 333).

**6.108.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 334).

**6.108.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 334).

**6.108.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 335).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h`

## 6.109 `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 710).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMa
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller`:

### Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.109.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 710). NOTE!  
 This file is auto generated - do not modify! if you need to make a change, please see  
 the Java Classes in the activemq-openwire-generator module

### 6.109.2 Constructor & Destructor Documentation

**6.109.2.1** `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller  
 ( ) [inline]`

**6.109.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller  
 ( ) [inline, virtual]`

### 6.109.3 Member Function Documentation

**6.109.3.1** `virtual commands::DataStructure* ac-  
 tivemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.109.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::getDataStructureType  
 ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.109.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::looseMarshal  
 ( OpenWireFormat * wireFormat, commands::DataStructure  
 * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 329).

**6.109.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 329).

**6.109.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 330).

**6.109.3.6** virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::tightMarshal2  
( OpenWireFormat \* *wireFormat*, commands::DataStructure  
\* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 330).

**6.109.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::tightUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream  
\* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 331).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h`

## 6.110 `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 714).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMa
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller`:

**Public Member Functions**

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.110.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 714). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.110.2 Constructor & Destructor Documentation

6.110.2.1 **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller**  
( ) [inline]

6.110.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller**  
( ) [inline, virtual]

### 6.110.3 Member Function Documentation

6.110.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.110.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

```
6.110.3.3  virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 337).

```
6.110.3.4  virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 337).

6.110.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 338).

6.110.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 338).

6.110.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 339).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h`

## 6.111 `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller` Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 718).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTopicMa
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller`:

#### Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.111.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 718). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.111.2 Constructor & Destructor Documentation

**6.111.2.1** **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller**  
( ) [inline]

**6.111.2.2** **virtual activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller**  
( ) [inline, virtual]

### 6.111.3 Member Function Documentation

**6.111.3.1** **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.111.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.111.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 345).

**6.111.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---



<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 345).

**6.111.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 346).

**6.111.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 346).

**6.111.3.7** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 347).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTopicMarshaller.h`

## 6.112 **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 722).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMa
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller**:

## Public Member Functions

- **ActiveMQTopicMarshaller ()**
- virtual **~ActiveMQTopicMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.112.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 722). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.112.2 Constructor & Destructor Documentation

- 6.112.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller ( ) [inline]`
- 6.112.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller ( ) [inline, virtual]`

### 6.112.3 Member Function Documentation

- 6.112.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

- 6.112.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

- 6.112.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 341).

**6.112.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 341).

**6.112.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 342).

```
6.112.3.6  virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 342).

```
6.112.3.7  virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 343).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h

## 6.113 activemq::core::ActiveMQTransactionContext Class Reference

Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.

```
#include <src/main/activemq/core/ActiveMQTransactionContext.h>
```

### Public Member Functions

- **ActiveMQTransactionContext** (**ActiveMQSession** \*session, const **decaf::util::Properties** &properties)

*Constructor.*

- virtual ~**ActiveMQTransactionContext** ()
- virtual void **addSynchronization** (const **Pointer**< **Synchronization** > &sync)

*Adds a **Synchronization** (p. 3826) to this Transaction.*

- virtual void **removeSynchronization** (const **Pointer**< **Synchronization** > &sync)

*Removes a **Synchronization** (p. 3826) to this Transaction.*

- virtual void **begin** () throw ( exceptions::ActiveMQException )

*Begins a new transaction if one is not currently in progress.*

- virtual void **commit** () throw ( exceptions::ActiveMQException )

*Commit the current Transaction.*

- virtual void **rollback** () throw ( exceptions::ActiveMQException )

*Rollback the current Transaction.*

- virtual const **decaf::lang::Pointer**< **commands::TransactionId** > & **getTransactionId** () const

*Get the Transaction Id object for the current Transaction, returns NULL if no transaction is running.*

- virtual bool **isInTransaction** () const

*Checks to see if there is currently a Transaction in progress returns false if not, true otherwise.*

### 6.113.1 Detailed Description

Transaction Management class, hold messages that are to be redelivered upon a request to roll-back. The Transaction represents an always running transaction, when it is committed or rolled back it silently creates a new transaction for the next set of messages. The only way to permanently end this transaction is to delete it.

Since

2.0

### 6.113.2 Constructor & Destructor Documentation

6.113.2.1 `activemq::core::ActiveMQTransactionContext::ActiveMQTransactionContext ( ActiveMQSession * session, const decaf::util::Properties & properties )`

Constructor.

Parameters

<i>session</i>	The session that contains this transaction
<i>properties</i>	Configuration parameters for this object

6.113.2.2 `virtual activemq::core::ActiveMQTransactionContext::~~ActiveMQTransactionContext ( ) [virtual]`

### 6.113.3 Member Function Documentation

6.113.3.1 `virtual void activemq::core::ActiveMQTransactionContext::addSynchronization ( const Pointer< Synchronization > & sync ) [virtual]`

Adds a **Synchronization** (p. 3826) to this Transaction.

Parameters

<i>sync</i>	- The <b>Synchronization</b> (p. 3826) instance to add.
-------------	---

6.113.3.2 `virtual void activemq::core::ActiveMQTransactionContext::begin ( ) throw ( exceptions::ActiveMQException ) [virtual]`

Begins a new transaction if one is not currently in progress.

Exceptions

<i>ActiveMQException</i>	
--------------------------	--



**6.113.3.3** `virtual void activemq::core::ActiveMQTransactionContext::commit ( ) throw ( exceptions::ActiveMQException ) [virtual]`

Commit the current Transaction.

#### Exceptions

<i>ActiveMQException</i>	
--------------------------	--

**6.113.3.4** `virtual const decaf::lang::Pointer<commands::TransactionId>& activemq::core::ActiveMQTransactionContext::getTransactionId ( ) const [virtual]`

Get the Transaction Id object for the current Transaction, returns NULL if no transaction is running.

#### Returns

TransactionInfo

#### Exceptions

<i>InvalidStateException</i>	if a Transaction is not in progress.
------------------------------	--------------------------------------

**6.113.3.5** `virtual bool activemq::core::ActiveMQTransactionContext::isInTransaction ( ) const [virtual]`

Checks to see if there is currently a Transaction in progress returns false if not, true otherwise.

#### Returns

true if a transaction is in progress.

**6.113.3.6** `virtual void activemq::core::ActiveMQTransactionContext::removeSynchronization ( const Pointer< Synchronization > & sync ) [virtual]`

Removes a **Synchronization** (p. 3826) to this Transaction.

#### Parameters

<i>sync</i>	- The <b>Synchronization</b> (p. 3826) instance to add.
-------------	---

6.113.3.7 virtual void `activemq::core::ActiveMQTransactionContext::rollback ( )` throw ( `exceptions::ActiveMQException` ) [virtual]

Rollback the current Transaction.

#### Exceptions

<code>ActiveMQException</code>
--------------------------------

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQTransactionContext.h`

## 6.114 decaf::util::zip::Adler32 Class Reference

Clas that can be used to compute an Adler-32 **Checksum** (p. 1174) for a data stream.

```
#include <src/main/decaf/util/zip/Adler32.h>
```

Inheritance diagram for `decaf::util::zip::Adler32`:

### Public Member Functions

- **Adler32** ()
- virtual **~Adler32** ()
- virtual long long **getValue** () const
- virtual void **reset** ()  
*Reset the checksum to its initial value.*
- virtual void **update** (const std::vector< unsigned char > &buffer)  
*Updates the current checksum with the specified vector of bytes.*
- virtual void **update** (const std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Updates the current checksum with the specified array of bytes.*
- virtual void **update** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Updates the current checksum with the specified array of bytes.*
- virtual void **update** (int byte)  
*Updates the current checksum with the specified byte value.*

### 6.114.1 Detailed Description

Clas that can be used to compute an Adler-32 **Checksum** (p. 1174) for a data stream. The Alder-32 checksum trades reliability for speed over the CRC-32 algorithm.

Since

1.0

### 6.114.2 Constructor & Destructor Documentation

6.114.2.1 `decaf::util::zip::Adler32 ( )`

6.114.2.2 `virtual decaf::util::zip::Adler32::~~Adler32 ( ) [virtual]`

### 6.114.3 Member Function Documentation

6.114.3.1 `virtual long long decaf::util::zip::Adler32::getValue ( ) const [virtual]`

**Returns**

the current checksum value.

Implements `decaf::util::zip::Checksum` (p. 1175).

6.114.3.2 `virtual void decaf::util::zip::Adler32::reset ( ) [virtual]`

Reset the checksum to its initial value.

Implements `decaf::util::zip::Checksum` (p. 1175).

6.114.3.3 `virtual void decaf::util::zip::Adler32::update ( const unsigned char * buffer, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Updates the current checksum with the specified array of bytes.

**Parameters**

<i>buffer</i>	The buffer to read the updated bytes from.
<i>size</i>	The size of the passed buffer.
<i>offset</i>	The position in the buffer to start reading.
<i>length</i>	The amount of data to read from the byte buffer.

**Exceptions**

<i>NullPointerException</i>	if the passed buffer is NULL.
-----------------------------	-------------------------------

<i>IndexOutOfBoundsException</i>	if offset + length > size of the buffer.
----------------------------------	--

Implements **decaf::util::zip::Checksum** (p. 1175).

**6.114.3.4** `virtual void decaf::util::zip::Adler32::update ( const std::vector< unsigned char > & buffer, int offset, int length ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Updates the current checksum with the specified array of bytes.

#### Parameters

<i>buffer</i>	The buffer to read the updated bytes from.
<i>offset</i>	The position in the buffer to start reading.
<i>length</i>	The amount of data to read from the byte buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if offset + length > size of the buffer.
----------------------------------	--

Implements **decaf::util::zip::Checksum** (p. 1176).

**6.114.3.5** `virtual void decaf::util::zip::Adler32::update ( const std::vector< unsigned char > & buffer )` [virtual]

Updates the current checksum with the specified vector of bytes.

#### Parameters

<i>buffer</i>	The buffer to read the updated bytes from.
---------------	--

Implements **decaf::util::zip::Checksum** (p. 1176).

**6.114.3.6** `virtual void decaf::util::zip::Adler32::update ( int byte )` [virtual]

Updates the current checksum with the specified byte value.

#### Parameters

<i>byte</i>	The byte value to update the current <b>Checksum</b> (p. 1174) with (0..255).
-------------	---

Implements **decaf::util::zip::Checksum** (p. 1176).

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**Adler32.h**

## 6.115 decaf::lang::Appendable Class Reference

An object to which char sequences and values can be appended.

```
#include <src/main/decaf/lang/Appendable.h>
```

Inheritance diagram for decaf::lang::Appendable:

### Public Member Functions

- virtual `~Appendable()`
- virtual `Appendable & append(char value)=0` throw ( decaf::lang::Exception )  
*Appends the specified character to this **Appendable** (p. 733).*
- virtual `Appendable & append(const CharSequence *csq)=0` throw ( decaf::lang::Exception )  
*Appends the specified character sequence to this **Appendable** (p. 733).*
- virtual `Appendable & append(const CharSequence *csq, int start, int end)=0` throw ( decaf::lang::Exception )  
*Appends a subsequence of the specified character sequence to this **Appendable** (p. 733).*

### 6.115.1 Detailed Description

An object to which char sequences and values can be appended. The **Appendable** (p. 733) interface must be implemented by any class whose instances are intended to receive formatted output from a Formatter.

TODO The characters to be appended should be valid Unicode characters as described in Unicode **Character** (p. 1126) Representation. Note that supplementary characters may be composed of multiple 16-bit char values.

Appendables are not necessarily safe for multithreaded access. **Thread** (p. 3876) safety is the responsibility of classes that extend and implement this interface.

Since this interface may be implemented by existing classes with different styles of error handling there is no guarantee that errors will be propagated to the invoker.

Since

1.0

## 6.115.2 Constructor & Destructor Documentation

6.115.2.1 `virtual decaf::lang::Appendable::~~Appendable ( )` [`inline`, `virtual`]

## 6.115.3 Member Function Documentation

6.115.3.1 `virtual Appendable& decaf::lang::Appendable::append ( char value ) throw ( decaf::lang::Exception )` [`pure virtual`]

Appends the specified character to this **Appendable** (p. 733).

### Parameters

<i>value</i>	The character to append.
--------------	--------------------------

### Returns

a Reference to this **Appendable** (p. 733)

### Exceptions

<i>Exception</i> (p. 1886)	if an error occurs.
----------------------------	---------------------

Implemented in **decaf::io::Writer** (p. 4135), and **decaf::nio::CharBuffer** (p. 1153).

6.115.3.2 `virtual Appendable& decaf::lang::Appendable::append ( const CharSequence * csq, int start, int end ) throw ( decaf::lang::Exception )` [`pure virtual`]

Appends a subsequence of the specified character sequence to this **Appendable** (p. 733).

### Parameters

<i>csq</i>	- The character sequence from which a subsequence will be appended. If <i>csq</i> is NULL, then characters will be appended as if <i>csq</i> contained the string "null".
<i>start</i>	The index of the first character in the subsequence.
<i>end</i>	The index of the character following the last character in the subsequence.

### Returns

a Reference to this **Appendable** (p. 733)

### Exceptions

<i>Exception</i> (p. 1886)	if an error occurs.
<i>IndexOutOfBoundsException</i>	<i>start</i> is greater than <i>end</i> , or <i>end</i> is greater than <i>csq.length()</i>

Implemented in **decaf::io::Writer** (p. 4136), and **decaf::nio::CharBuffer** (p. 1153).

6.115.3.3 virtual **Appendable**& decaf::lang::Appendable::append ( const CharSequence \* csq ) throw ( decaf::lang::Exception ) [pure virtual]

Appends the specified character sequence to this **Appendable** (p. 733).

#### Parameters

<i>csq</i>	The character sequence from which a subsequence will be appended. If csq is NULL, then characters will be appended as if csq contained the string "null".
------------	---

#### Returns

a Reference to this **Appendable** (p. 733).

#### Exceptions

<i>Exception</i> (p. 1886)	if an error occurs.
----------------------------	---------------------

Implemented in **decaf::io::Writer** (p. 4136), and **decaf::nio::CharBuffer** (p. 1154).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Appendable.h**

## 6.116 decaf::internal::AprPool Class Reference

Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.

```
#include <src/main/decaf/internal/AprPool.h>
```

#### Public Member Functions

- **AprPool** ()
- virtual ~**AprPool** ()
- apr\_pool\_t \* **getAprPool** () const  
*Gets the internal APR Pool.*
- void **cleanup** ()  
*Clears data that was allocated by this pool.*

#### Static Public Member Functions

- static apr\_pool\_t \* **getGlobalPool** ()  
*Gets a pointer to the Global APR Pool used for the Application.*

### 6.116.1 Detailed Description

Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.

### 6.116.2 Constructor & Destructor Documentation

6.116.2.1 `decaf::internal::AprPool::AprPool ( )`

6.116.2.2 `virtual decaf::internal::AprPool::~~AprPool ( )` [virtual]

### 6.116.3 Member Function Documentation

6.116.3.1 `void decaf::internal::AprPool::cleanup ( )`

Clears data that was allocated by this pool.

Users should call this after getting the data from the APR functions and copying it to someplace safe.

6.116.3.2 `apr_pool_t* decaf::internal::AprPool::getAprPool ( ) const`

Gets the internal APR Pool.

#### Returns

the internal APR pool

6.116.3.3 `static apr_pool_t* decaf::internal::AprPool::getGlobalPool ( )` [static]

Gets a pointer to the Global APR Pool used for the Application.

#### Returns

pointer to the global APR Pool

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/AprPool.h`

## 6.117 `decaf::lang::ArrayPointer< T, REFCOUNTER >` Class Template Reference

Decaf's implementation of a Smart **Pointer** (p. 3032) that is a template on a Type and is **Thread** (p. 3876) Safe if the default Reference Counter is used.

```
#include <src/main/decaf/lang/ArrayPointer.h>
```



## Data Structures

- struct **ArrayData**

## Public Types

- typedef T \* **PointerType**
- typedef T & **ReferenceType**
- typedef const T & **ConstReferenceType**
- typedef REFCOUNTER **CounterType**

## Public Member Functions

- **ArrayPointer** ()  
*Default Constructor.*
- **ArrayPointer** (int size)  
*Create a new **ArrayPointer** (p. 736) instance and allocates an internal array that is sized using the passed in size value.*
- **ArrayPointer** (const **PointerType** value, int size)  
*Explicit Constructor, creates an **ArrayPointer** (p. 736) that contains value with a single reference.*
- **ArrayPointer** (const **ArrayPointer** &value) throw ()  
*Copy constructor.*
- virtual ~**ArrayPointer** () throw ()
- void **reset** (T \*value, int size=0)  
*Resets the **ArrayPointer** (p. 736) to hold the new value.*
- T \* **release** ()  
*Releases the **Pointer** (p. 3032) held and resets the internal pointer value to Null.*
- **PointerType** **get** () const  
*Gets the real array pointer that is contained within this **Pointer** (p. 3032).*
- int **length** () const  
*Returns the current size of the contained array or zero if the array is NULL.*
- void **swap** (**ArrayPointer** &value) throw ()  
*Exception (p. 1886) Safe Swap Function.*
- **ArrayPointer** **clone** () const  
*Creates a new **ArrayPointer** (p. 736) instance that is a clone of the value contained in this **ArrayPointer** (p. 736).*

- **ArrayPointer & operator=** (const **ArrayPointer** &right) throw ()

*Assigns the value of right to this **Pointer** (p. 3032) and increments the reference Count.*

- template<typename T1 , typename R1 >  
**ArrayPointer & operator=** (const **ArrayPointer**< T1, R1 > &right) throw ()
- **ReferenceType operator[]** (int index)

*Dereference Operator, returns a reference to the Contained value.*

- **ConstReferenceType operator[]** (int index) const
- bool **operator!** () const
- template<typename T1 , typename R1 >  
 bool **operator==** (const **ArrayPointer**< T1, R1 > &right) const
- template<typename T1 , typename R1 >  
 bool **operator!=** (const **ArrayPointer**< T1, R1 > &right) const

## Friends

- bool **operator==** (const **ArrayPointer** &left, const T \*right)
- bool **operator==** (const T \*left, const **ArrayPointer** &right)
- bool **operator!=** (const **ArrayPointer** &left, const T \*right)
- bool **operator!=** (const T \*left, const **ArrayPointer** &right)

### 6.117.1 Detailed Description

```
template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter>
class decaf::lang::ArrayPointer< T, REFCOUNTER >
```

Decaf's implementation of a Smart **Pointer** (p. 3032) that is a template on a Type and is **Thread** (p. 3876) Safe if the default Reference Counter is used. This **Pointer** (p. 3032) type allows for the substitution of different Reference Counter implementations which provide a means of using invasive reference counting if desired using a custom implementation of `ReferenceCounter`.

The Decaf smart pointer provide comparison operators for comparing **Pointer** (p. 3032) instances in the same manner as normal pointer, except that it does not provide an overload of operators ( <, <=, >, >= ). To allow use of a **Pointer** (p. 3032) in a STL container that requires it, **Pointer** (p. 3032) provides an implementation of `std::less`.

## Since

1.0

## 6.117.2 Member Typedef Documentation

6.117.2.1 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> typedef const T&  
decaf::lang::ArrayPointer< T, REFCOUNTER >::ConstReferenceType`

6.117.2.2 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> typedef REFCOUNTER  
decaf::lang::ArrayPointer< T, REFCOUNTER >::CounterType`

6.117.2.3 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> typedef T*  
decaf::lang::ArrayPointer< T, REFCOUNTER >::PointerType`

6.117.2.4 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> typedef T&  
decaf::lang::ArrayPointer< T, REFCOUNTER >::ReferenceType`

## 6.117.3 Constructor & Destructor Documentation

6.117.3.1 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter>  
decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayPointer ( )  
[inline]`

Default Constructor.

Initialized the contained array pointer to NULL, using the subscript operator results in an exception unless reset to contain a real value.

Referenced by `decaf::lang::ArrayPointer< unsigned char >::clone()`, and `decaf::lang::ArrayPointer< unsigned char >::reset()`.

6.117.3.2 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter>  
decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayPointer ( int size )  
[inline]`

Create a new **ArrayPointer** (p. 736) instance and allocates an internal array that is sized using the passed in size value.

### Parameters

<i>size</i>	The size of the array to allocate for this <b>ArrayPointer</b> (p. 736) instance.
-------------	---

```
6.117.3.3  template<typename T, typename REFCOUNTER =
            decaf::util::concurrent::atomic::AtomicRefCounter>
            decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayPointer ( const
            PointerType value, int size ) [inline, explicit]
```

Explicit Constructor, creates an **ArrayPointer** (p. 736) that contains value with a single reference.

This object now has ownership until a call to release.

#### Parameters

<i>value</i>	The pointer to the instance of the array we are taking ownership of.
<i>size</i>	The size of the array this object is taking ownership of.

```
6.117.3.4  template<typename T, typename REFCOUNTER =
            decaf::util::concurrent::atomic::AtomicRefCounter>
            decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayPointer ( const
            ArrayPointer< T, REFCOUNTER > & value ) throw () [inline]
```

Copy constructor.

Copies the value contained in the **ArrayPointer** (p. 736) to the new instance and increments the reference counter.

```
6.117.3.5  template<typename T, typename REFCOUNTER =
            decaf::util::concurrent::atomic::AtomicRefCounter> virtual
            decaf::lang::ArrayPointer< T, REFCOUNTER >::~~ArrayPointer ( ) throw ()
            [inline, virtual]
```

#### 6.117.4 Member Function Documentation

```
6.117.4.1  template<typename T, typename REFCOUNTER =
            decaf::util::concurrent::atomic::AtomicRefCounter> ArrayPointer
            decaf::lang::ArrayPointer< T, REFCOUNTER >::clone ( ) const
            [inline]
```

Creates a new **ArrayPointer** (p. 736) instance that is a clone of the value contained in this **ArrayPointer** (p. 736).

#### Returns

an **ArrayPointer** (p. 736) that contains a copy of the data in this **ArrayPointer** (p. 736).

```
6.117.4.2  template<typename T, typename REFCOUNTER =
            decaf::util::concurrent::atomic::AtomicRefCounter> PointerType
            decaf::lang::ArrayPointer< T, REFCOUNTER >::get ( ) const  [inline]
```

Gets the real array pointer that is contained within this **Pointer** (p. 3032).

This is not really safe since the caller could delete or alter the pointer but it mimics the STL `auto_ptr` and gives access in cases where the caller absolutely needs the real **Pointer** (p. 3032). Use at your own risk.

#### Returns

the contained pointer.

Referenced by `decaf::lang::ArrayPointer< unsigned char >::clone()`, `decaf::lang::ArrayPointerComparator< T, R >::compare()`, `decaf::lang::operator!=()`, `decaf::lang::ArrayPointer< unsigned char >::operator!=()`, `std::less< decaf::lang::ArrayPointer< T > >::operator()`, `decaf::lang::ArrayPointerComparator< T, R >::operator()`, `decaf::lang::operator==()`, and `decaf::lang::ArrayPointer< unsigned char >::operator==()`.

```
6.117.4.3  template<typename T, typename REFCOUNTER =
            decaf::util::concurrent::atomic::AtomicRefCounter> int
            decaf::lang::ArrayPointer< T, REFCOUNTER >::length ( ) const
            [inline]
```

Returns the current size of the contained array or zero if the array is NULL.

#### Returns

the size of the array or zero if the array is NULL

- 6.117.4.4 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> bool  
decaf::lang::ArrayPointer< T, REFCOUNTER >::operator! ( ) const  
[inline]`
- 6.117.4.5 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 ,  
typename R1 > bool decaf::lang::ArrayPointer< T, REFCOUNTER  
>::operator!= ( const ArrayPointer< T1, R1 > & right ) const [inline]`
- 6.117.4.6 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 ,  
typename R1 > ArrayPointer& decaf::lang::ArrayPointer< T, REFCOUNTER  
>::operator= ( const ArrayPointer< T1, R1 > & right ) throw () [inline]`
- 6.117.4.7 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> ArrayPointer&  
decaf::lang::ArrayPointer< T, REFCOUNTER >::operator= ( const  
ArrayPointer< T, REFCOUNTER > & right ) throw () [inline]`

Assigns the value of *right* to this **Pointer** (p. 3032) and increments the reference Count.

#### Parameters

<i>right</i>	- <b>Pointer</b> (p. 3032) on the right hand side of an operator= call to this.
--------------	---

- 6.117.4.8 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 ,  
typename R1 > bool decaf::lang::ArrayPointer< T, REFCOUNTER  
>::operator== ( const ArrayPointer< T1, R1 > & right ) const [inline]`
- 6.117.4.9 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> ReferenceType  
decaf::lang::ArrayPointer< T, REFCOUNTER >::operator[] ( int index )  
[inline]`

Dereference Operator, returns a reference to the Contained value.

This method throws an `NullPointerException` if the contained value is `NULL`.

#### Returns

reference to the contained pointer.

#### Exceptions

<code>NullPointerException</code>	if the contained value is <code>Null</code>
-----------------------------------	---

```
6.117.4.10 template<typename T, typename REFCOUNTER =
    decaf::util::concurrent::atomic::AtomicRefCount> ConstReferenceType
    decaf::lang::ArrayPointer< T, REFCOUNTER >::operator[] ( int index ) const
    [inline]
```

```
6.117.4.11 template<typename T, typename REFCOUNTER =
    decaf::util::concurrent::atomic::AtomicRefCount> T*
    decaf::lang::ArrayPointer< T, REFCOUNTER >::release ( ) [inline]
```

Releases the **Pointer** (p. 3032) held and resets the internal pointer value to Null.

This method is not guaranteed to be safe if the **Pointer** (p. 3032) is held by more than one object or this method is called from more than one thread.

#### Parameters

<i>value</i>	- The new value to contain.
--------------	-----------------------------

#### Returns

The pointer instance that was held by this **Pointer** (p. 3032) object, the pointer is no longer owned by this **Pointer** (p. 3032) and won't be freed when this **Pointer** (p. 3032) goes out of scope.

Referenced by decaf::lang::ArrayPointer< unsigned char >::ArrayPointer().

```
6.117.4.12 template<typename T, typename REFCOUNTER =
    decaf::util::concurrent::atomic::AtomicRefCount> void
    decaf::lang::ArrayPointer< T, REFCOUNTER >::reset ( T * value, int size = 0
    ) [inline]
```

Resets the **ArrayPointer** (p. 736) to hold the new value.

Before the new value is stored reset checks if the old value should be destroyed and if so calls delete. Call reset with a value of NULL is supported and acts to set this **Pointer** (p. 3032) to a NULL pointer.

#### Parameters

<i>value</i>	The new array pointer value to contain.
<i>size</i>	The size of the new array value this object now contains.

```
6.117.4.13 template<typename T, typename REFCOUNTER =
    decaf::util::concurrent::atomic::AtomicRefCount> void
    decaf::lang::ArrayPointer< T, REFCOUNTER >::swap ( ArrayPointer< T,
    REFCOUNTER > & value ) throw () [inline]
```

**Exception** (p. 1886) Safe Swap Function.

**Parameters**

<i>value</i>	- the value to swap with this.
--------------	--------------------------------

Referenced by `decaf::lang::ArrayPointer< unsigned char >::operator=()`, and `decaf::lang::ArrayPointer< unsigned char >::swap()`.

**6.117.5 Friends And Related Function Documentation**

- 6.117.5.1 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool operator!= ( const ArrayPointer< T, REFCOUNTER > & left, const T * right ) [friend]`
- 6.117.5.2 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool operator!= ( const T * left, const ArrayPointer< T, REFCOUNTER > & right ) [friend]`
- 6.117.5.3 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool operator== ( const ArrayPointer< T, REFCOUNTER > & left, const T * right ) [friend]`
- 6.117.5.4 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool operator== ( const T * left, const ArrayPointer< T, REFCOUNTER > & right ) [friend]`

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/ArrayPointer.h`

**6.118 decaf::lang::ArrayPointerComparator< T, R > Class Template Reference**

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this **ArrayPointer** (p. 736).

```
#include <src/main/decaf/lang/ArrayPointer.h>
```

Inheritance diagram for `decaf::lang::ArrayPointerComparator< T, R >`:

**Public Member Functions**

- virtual bool **operator()** (const **ArrayPointer**< T, R > &left, const **ArrayPointer**< T, R > &right) const
- virtual int **compare** (const **ArrayPointer**< T, R > &left, const **ArrayPointer**< T, R > &right) const



### 6.118.1 Detailed Description

```
template<typename T, typename R = decaf::util::concurrent::atomic::AtomicRefCounter> class
decaf::lang::ArrayPointerComparator< T, R >
```

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this **ArrayPointer** (p. 736). This allows for a basic ordering to be acheived in Decaf containers.

Custom implementations are possible where an array of some type has a logical natural ordering such as array of integers where the sum of all ints in the array is used.

### 6.118.2 Member Function Documentation

```
6.118.2.1 template<typename T , typename R =
decaf::util::concurrent::atomic::AtomicRefCounter> virtual int
decaf::lang::ArrayPointerComparator< T, R >::compare ( const
ArrayPointer< T, R > & left, const ArrayPointer< T, R > & right ) const
[inline, virtual]
```

References decaf::lang::ArrayPointer< T, REFCOUNTER >::get().

```
6.118.2.2 template<typename T , typename R =
decaf::util::concurrent::atomic::AtomicRefCounter> virtual bool
decaf::lang::ArrayPointerComparator< T, R >::operator() ( const
ArrayPointer< T, R > & left, const ArrayPointer< T, R > & right ) const
[inline, virtual]
```

References decaf::lang::ArrayPointer< T, REFCOUNTER >::get().

The documentation for this class was generated from the following file:

- src/main/decaf/lang/ArrayPointer.h

## 6.119 decaf::util::concurrent::atomic::AtomicBoolean Class Reference

A boolean value that may be updated atomically.

```
#include <src/main/decaf/util/concurrent/atomic/AtomicBoolean.h>
```

### Public Member Functions

- **AtomicBoolean ()**  
*Creates a new **AtomicBoolean** (p. 745) whose initial value is false.*

- **AtomicBoolean** (bool initialValue)  
*Creates a new **AtomicBoolean** (p. 745) with the initial value.*
- virtual **~AtomicBoolean** ()
- bool **get** () const  
*Gets the current value of this **AtomicBoolean** (p. 745).*
- void **set** (bool newValue)  
*Unconditionally sets to the given value.*
- bool **compareAndSet** (bool expect, bool update)  
*Atomically sets the value to the given updated value if the current value == the expected value.*
- bool **getAndSet** (bool newValue)  
*Atomically sets to the given value and returns the previous value.*
- std::string **toString** () const  
*Returns the String representation of the current value.*

### 6.119.1 Detailed Description

A boolean value that may be updated atomically. An **AtomicBoolean** (p. 745) is used in applications such as atomically updated flags, and cannot be used as a replacement for a Boolean.

### 6.119.2 Constructor & Destructor Documentation

#### 6.119.2.1 `decaf::util::concurrent::atomic::AtomicBoolean::AtomicBoolean ( )`

Creates a new **AtomicBoolean** (p. 745) whose initial value is false.

#### 6.119.2.2 `decaf::util::concurrent::atomic::AtomicBoolean::AtomicBoolean ( bool initialValue )`

Creates a new **AtomicBoolean** (p. 745) with the initial value.

#### Parameters

<i>initialValue</i>	- The initial value of this boolean.
---------------------	--------------------------------------

6.119.2.3 `virtual decaf::util::concurrent::atomic::AtomicBoolean::~~AtomicBoolean ( )`  
[inline, virtual]

### 6.119.3 Member Function Documentation

6.119.3.1 `bool decaf::util::concurrent::atomic::AtomicBoolean::compareAndSet ( bool expect, bool update )`

Atomically sets the value to the given updated value if the current value == the expected value.

#### Parameters

<i>expect</i>	- the expected value
<i>update</i>	- the new value

#### Returns

true if successful. False return indicates that the actual value was not equal to the expected value.

6.119.3.2 `bool decaf::util::concurrent::atomic::AtomicBoolean::get ( ) const` [inline]

Gets the current value of this **AtomicBoolean** (p. 745).

#### Returns

the currently set value.

6.119.3.3 `bool decaf::util::concurrent::atomic::AtomicBoolean::getAndSet ( bool newValue )`

Atomically sets to the given value and returns the previous value.

#### Parameters

<i>newValue</i>	- the new value
-----------------	-----------------

#### Returns

the previous value

6.119.3.4 `void decaf::util::concurrent::atomic::AtomicBoolean::set ( bool newValue )`  
[inline]

Unconditionally sets to the given value.

**Parameters**

<i>newValue</i>	- the new value
-----------------	-----------------

**6.119.3.5 std::string decaf::util::concurrent::atomic::AtomicBoolean::toString ( ) const**

Returns the String representation of the current value.

**Returns**

the String representation of the current value.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/atomic/**AtomicBoolean.h**

**6.120 decaf::util::concurrent::atomic::AtomicInteger Class Reference**

An int value that may be updated atomically.

```
#include <src/main/decaf/util/concurrent/atomic/AtomicInteger.h>
```

Inheritance diagram for decaf::util::concurrent::atomic::AtomicInteger:

**Public Member Functions**

- **AtomicInteger ( )**  
*Create a new **AtomicInteger** (p. 748) with an initial value of 0.*
- **AtomicInteger (int initialValue)**  
*Create a new **AtomicInteger** (p. 748) with the given initial value.*
- virtual **~AtomicInteger ( )**
- int **get ( ) const**  
*Gets the current value.*
- void **set (int newValue)**  
*Sets to the given value.*
- int **getAndSet (int newValue)**  
*Atomically sets to the given value and returns the old value.*
- bool **compareAndSet (int expect, int update)**  
*Atomically sets the value to the given updated value if the current value == the expected value.*

- **int getAndIncrement ()**  
*Atomically increments by one the current value.*
- **int getAndDecrement ()**  
*Atomically decrements by one the current value.*
- **int getAndAdd (int delta)**  
*Atomically adds the given value to the current value.*
- **int incrementAndGet ()**  
*Atomically increments by one the current value.*
- **int decrementAndGet ()**  
*Atomically decrements by one the current value.*
- **int addAndGet (int delta)**  
*Atomically adds the given value to the current value.*
- **std::string toString () const**  
*Returns the String representation of the current value.*
- **int intValue () const**  
*Description copied from class: Number Returns the value of the specified number as an int.*
- **long long longValue () const**  
*Description copied from class: Number Returns the value of the specified number as a long.*
- **float floatValue () const**  
*Description copied from class: Number Returns the value of the specified number as a float.*
- **double doubleValue () const**  
*Description copied from class: Number Returns the value of the specified number as a double.*

### 6.120.1 Detailed Description

An int value that may be updated atomically. An **AtomicInteger** (p. 748) is used in applications such as atomically incremented counters, and cannot be used as a replacement for an Integer. However, this class does extend Number to allow uniform access by tools and utilities that deal with numerically-based classes.

## 6.120.2 Constructor & Destructor Documentation

### 6.120.2.1 `decaf::util::concurrent::atomic::AtomicInteger::AtomicInteger ( )`

Create a new **AtomicInteger** (p. 748) with an initial value of 0.

### 6.120.2.2 `decaf::util::concurrent::atomic::AtomicInteger::AtomicInteger ( int initialValue )`

Create a new **AtomicInteger** (p. 748) with the given initial value.

#### Parameters

<i>initialValue</i>	- The initial value of this object.
---------------------	-------------------------------------

### 6.120.2.3 `virtual decaf::util::concurrent::atomic::AtomicInteger::~~AtomicInteger ( )` [inline, virtual]

## 6.120.3 Member Function Documentation

### 6.120.3.1 `int decaf::util::concurrent::atomic::AtomicInteger::addAndGet ( int delta )`

Atomically adds the given value to the current value.

#### Parameters

<i>delta</i>	- the value to add.
--------------	---------------------

#### Returns

the updated value.

### 6.120.3.2 `bool decaf::util::concurrent::atomic::AtomicInteger::compareAndSet ( int expect, int update )`

Atomically sets the value to the given updated value if the current value == the expected value.

#### Parameters

<i>expect</i>	- the expected value
<i>update</i>	- the new value

#### Returns

true if successful. False return indicates that the actual value was not equal to the expected value.

**6.120.3.3** `int decaf::util::concurrent::atomic::AtomicInteger::decrementAndGet ( )`

Atomically decrements by one the current value.

**Returns**

the updated value.

Referenced by `decaf::util::concurrent::atomic::AtomicRefCounter::release()`.

**6.120.3.4** `double decaf::util::concurrent::atomic::AtomicInteger::doubleValue ( ) const`  
[virtual]

Description copied from class: `Number` Returns the value of the specified number as a double.

This may involve rounding.

**Returns**

the numeric value represented by this object after conversion to type double.

Implements `decaf::lang::Number` (p. 2919).

**6.120.3.5** `float decaf::util::concurrent::atomic::AtomicInteger::floatValue ( ) const`  
[virtual]

Description copied from class: `Number` Returns the value of the specified number as a float.

This may involve rounding.

**Returns**

the numeric value represented by this object after conversion to type float.

Implements `decaf::lang::Number` (p. 2920).

**6.120.3.6** `int decaf::util::concurrent::atomic::AtomicInteger::get ( ) const` [inline]

Gets the current value.

**Returns**

the current value.

**6.120.3.7** `int decaf::util::concurrent::atomic::AtomicInteger::getAndAdd ( int delta )`

Atomically adds the given value to the current value.

**Parameters**

<i>delta</i>	- The value to add.
--------------	---------------------

**Returns**

the previous value.

**6.120.3.8 int decaf::util::concurrent::atomic::AtomicInteger::getAndDecrement ( )**

Atomically decrements by one the current value.

**Returns**

the previous value.

**6.120.3.9 int decaf::util::concurrent::atomic::AtomicInteger::getAndIncrement ( )**

Atomically increments by one the current value.

**Returns**

the previous value.

**6.120.3.10 int decaf::util::concurrent::atomic::AtomicInteger::getAndSet ( int *newValue* )**

Atomically sets to the given value and returns the old value.

**Parameters**

<i>newValue</i>	- the new value.
-----------------	------------------

**Returns**

the previous value.

**6.120.3.11 int decaf::util::concurrent::atomic::AtomicInteger::incrementAndGet ( )**

Atomically increments by one the current value.

**Returns**

the updated value.

Referenced by decaf::util::concurrent::atomic::AtomicRefCounter::AtomicRefCounter().



**6.120.3.12** `int decaf::util::concurrent::atomic::AtomicInteger::intValue ( ) const`  
[virtual]

Description copied from class: Number Returns the value of the specified number as an int.

This may involve rounding or truncation.

#### Returns

the numeric value represented by this object after conversion to type int.

Implements **decaf::lang::Number** (p. 2920).

**6.120.3.13** `long long decaf::util::concurrent::atomic::AtomicInteger::longValue ( ) const`  
[virtual]

Description copied from class: Number Returns the value of the specified number as a long.

This may involve rounding or truncation.

#### Returns

the numeric value represented by this object after conversion to type long long.

Implements **decaf::lang::Number** (p. 2920).

**6.120.3.14** `void decaf::util::concurrent::atomic::AtomicInteger::set ( int newValue )`  
[inline]

Sets to the given value.

#### Parameters

<i>newValue</i>	- the new value
-----------------	-----------------

**6.120.3.15** `std::string decaf::util::concurrent::atomic::AtomicInteger::toString ( ) const`

Returns the String representation of the current value.

#### Returns

the String representation of the current value.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/atomic/AtomicInteger.h`

## 6.121 decaf::util::concurrent::atomic::AtomicRefCounter Class Reference

```
#include <src/main/decaf/util/concurrent/atomic/AtomicRefCounter.h>
```

Inherited by `decaf::lang::ArrayPointer< unsigned char >`, `decaf::lang::Pointer< activemq::threads::TaskRunner >`, `decaf::lang::Pointer< ActiveMQDestination >`, `decaf::lang::Pointer< ActiveMQTransactionContext >`, `decaf::lang::Pointer< BackupTransportPool >`, `decaf::lang::Pointer< BooleanExpression >`, `decaf::lang::Pointer< BrokerError >`, `decaf::lang::Pointer< BrokerId >`, `decaf::lang::Pointer< ByteArrayAdapter >`, `decaf::lang::Pointer< CloseTransportsTask >`, `decaf::lang::Pointer< cms::Destination >`, `decaf::lang::Pointer< commands::ActiveMQDestination >`, `decaf::lang::Pointer< commands::ConsumerId >`, `decaf::lang::Pointer< commands::ConsumerInfo >`, `decaf::lang::Pointer< commands::Message >`, `decaf::lang::Pointer< commands::MessageAck >`, `decaf::lang::Pointer< commands::ProducerInfo >`, `decaf::lang::Pointer< commands::SessionInfo >`, `decaf::lang::Pointer< commands::TransactionId >`, `decaf::lang::Pointer< commands::WireFormatInfo >`, `decaf::lang::Pointer< Comparator< E > >`, `decaf::lang::Pointer< CompositeTaskRunner >`, `decaf::lang::Pointer< ConnectionId >`, `decaf::lang::Pointer< ConnectionInfo >`, `decaf::lang::Pointer< ConsumerId >`, `decaf::lang::Pointer< ConsumerInfo >`, `decaf::lang::Pointer< core::ActiveMQAckHandler >`, `decaf::lang::Pointer< DataStructure >`, `decaf::lang::Pointer< decaf::lang::Runnable >`, `decaf::lang::Pointer< decaf::lang::Thread >`, `decaf::lang::Pointer< Exception >`, `decaf::lang::Pointer< LockHandle >`, `decaf::lang::Pointer< LogManagerInternals >`, `decaf::lang::Pointer< Message >`, `decaf::lang::Pointer< MessageAck >`, `decaf::lang::Pointer< MessageId >`, `decaf::lang::Pointer< ProducerId >`, `decaf::lang::Pointer< ProducerInfo >`, `decaf::lang::Pointer< Properties >`, `decaf::lang::Pointer< Response >`, `decaf::lang::Pointer< ResponseBuilder >`, `decaf::lang::Pointer< SessionId >`, `decaf::lang::Pointer< SessionInfo >`, `decaf::lang::Pointer< Tracked >`, `decaf::lang::Pointer< TransactionId >`, `decaf::lang::Pointer< TransactionState >`, `decaf::lang::Pointer< Transport >`, `decaf::lang::Pointer< TransportListener >`, `decaf::lang::Pointer< URI >`, `decaf::lang::Pointer< URIPool >`, and `decaf::lang::Pointer< wireformat::WireFormat >`.

### Public Member Functions

- `AtomicRefCounter ()`
- `AtomicRefCounter (const AtomicRefCounter &other)`
- `virtual ~AtomicRefCounter ()`

### Protected Member Functions

- `void swap (AtomicRefCounter &other)`

*Swaps this instance's reference counter with the one given, this allows for copy-and-swap semantics of this object.*

- `bool release ()`

*Removes a reference to the counter Atomically and returns if the counter has reached zero, once the counter hits zero, the internal counter is destroyed and this instance is now considered to be unreferenced.*

### 6.121.1 Constructor & Destructor Documentation

**6.121.1.1** `decaf::util::concurrent::atomic::AtomicRefCounter::AtomicRefCounter ( )`  
[inline]

**6.121.1.2** `decaf::util::concurrent::atomic::AtomicRefCounter::AtomicRefCounter ( const AtomicRefCounter & other )` [inline]

References `decaf::util::concurrent::atomic::AtomicInteger::incrementAndGet()`.

**6.121.1.3** `virtual decaf::util::concurrent::atomic::AtomicRefCounter::~~AtomicRefCounter ( )`  
[inline, virtual]

### 6.121.2 Member Function Documentation

**6.121.2.1** `bool decaf::util::concurrent::atomic::AtomicRefCounter::release ( )` [inline, protected]

Removes a reference to the counter Atomically and returns if the counter has reached zero, once the counter hits zero, the internal counter is destroyed and this instance is now considered to be unreferenced.

#### Returns

true if the count is now zero.

Reimplemented in `decaf::lang::ArrayPointer< unsigned char >` (p. 743), `decaf::lang::Pointer< MessageAck >` (p. 3039), `decaf::lang::Pointer< BooleanExpression >` (p. 3039), `decaf::lang::Pointer< commands::ConsumerId >` (p. 3039), `decaf::lang::Pointer< BrokerError >` (p. 3039), `decaf::lang::Pointer< Transport >` (p. 3039), `decaf::lang::Pointer< wireformat::WireFormat >` (p. 3039), `decaf::lang::Pointer< commands::WireFormatInfo >` (p. 3039), `decaf::lang::Pointer< CloseTransportsTask >` (p. 3039), `decaf::lang::Pointer< CompositeTaskRunner >` (p. 3039), `decaf::lang::Pointer< ActiveMQTransactionContext >` (p. 3039), `decaf::lang::Pointer< commands::ProducerInfo >` (p. 3039), `decaf::lang::Pointer< Comparator< E > >` (p. 3039), `decaf::lang::Pointer< BrokerId >` (p. 3039), `decaf::lang::Pointer< LogManagerInternals >` (p. 3039), `decaf::lang::Pointer< commands::SessionInfo >` (p. 3039), `decaf::lang::Pointer< Message >` (p. 3039), `decaf::lang::Pointer< URI >` (p. 3039), `decaf::lang::Pointer< DataStructure >` (p. 3039), `decaf::lang::Pointer< activemq::threads::TaskRunner >` (p. 3039), `decaf::lang::Pointer< LockHandle >` (p. 3039), `decaf::lang::Pointer< commands::ActiveMQDestination >` (p. 3039), `decaf::lang::Pointer< ConsumerInfo >` (p. 3039), `decaf::lang::Pointer< ConnectionId >` (p. 3039), `decaf::lang::Pointer< decaf::lang::Runnable >` (p. 3039), `decaf::lang::Pointer< Properties >` (p. 3039), `decaf::lang::Pointer< BackupTransportPool >` (p. 3039), `decaf::lang::Pointer< ProducerInfo >` (p. 3039), `decaf::lang::Pointer<`

decaf::lang::Thread > (p. 3039), decaf::lang::Pointer< MessageId > (p. 3039), decaf::lang::Pointer< Response > (p. 3039), decaf::lang::Pointer< SessionId > (p. 3039), decaf::lang::Pointer< cms::Destination > (p. 3039), decaf::lang::Pointer< TransportListener > (p. 3039), decaf::lang::Pointer< commands::TransactionId > (p. 3039), decaf::lang::Pointer< ActiveMQDestination > (p. 3039), decaf::lang::Pointer< ProducerId > (p. 3039), decaf::lang::Pointer< ResponseBuilder > (p. 3039), decaf::lang::Pointer< SessionInfo > (p. 3039), decaf::lang::Pointer< commands::Message > (p. 3039), decaf::lang::Pointer< Tracked > (p. 3039), decaf::lang::Pointer< ConnectionInfo > (p. 3039), decaf::lang::Pointer< commands::MessageAck > (p. 3039), decaf::lang::Pointer< core::ActiveMQAckHandler > (p. 3039), decaf::lang::Pointer< Exception > (p. 3039), decaf::lang::Pointer< TransactionState > (p. 3039), decaf::lang::Pointer< commands::ConsumerInfo > (p. 3039), decaf::lang::Pointer< ConsumerId > (p. 3039), decaf::lang::Pointer< URIPool > (p. 3039), decaf::lang::Pointer< ByteArrayAdapter > (p. 3039), and decaf::lang::Pointer< TransactionId > (p. 3039).

References decaf::util::concurrent::atomic::AtomicInteger::decrementAndGet().

**6.121.2.2** void decaf::util::concurrent::atomic::AtomicRefCounter::swap (AtomicRefCounter & other ) [inline, protected]

Swaps this instance's reference counter with the one given, this allows for copy-and-swap semantics of this object.

#### Parameters

<i>other</i>	The value to swap with this one's.
--------------	------------------------------------

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/atomic/AtomicRefCounter.h

## 6.122 decaf::util::concurrent::atomic::AtomicReference< T > Class Template Reference

An Pointer reference that may be updated atomically.

```
#include <src/main/decaf/util/concurrent/atomic/AtomicReference.h>
```

#### Public Member Functions

- AtomicReference ()
- AtomicReference (T \*value)
- virtual ~AtomicReference ()
- T \* get () const  
*Gets the Current Value.*
- void set (T \*newValue)

*Sets the Current value of this Reference.*

- **bool compareAndSet** (T \*expect, T \*update)  
*Atomically sets the value to the given updated value if the current value == the expected value.*
- **T \*getAndSet** (T \*newValue)  
*Atomically sets to the given value and returns the old value.*
- **std::string toString** () const  
*Returns the String representation of the current value.*

### 6.122.1 Detailed Description

`template<typename T> class decaf::util::concurrent::atomic::AtomicReference< T >`

An Pointer reference that may be updated atomically.

### 6.122.2 Constructor & Destructor Documentation

**6.122.2.1** `template<typename T> decaf::util::concurrent::atomic::AtomicReference< T >::AtomicReference ( ) [inline]`

**6.122.2.2** `template<typename T> decaf::util::concurrent::atomic::AtomicReference< T >::AtomicReference ( T *value ) [inline]`

**6.122.2.3** `template<typename T> virtual decaf::util::concurrent::atomic::AtomicReference< T >::~~AtomicReference ( ) [inline, virtual]`

### 6.122.3 Member Function Documentation

**6.122.3.1** `template<typename T> bool decaf::util::concurrent::atomic::AtomicReference< T >::compareAndSet ( T * expect, T * update ) [inline]`

Atomically sets the value to the given updated value if the current value == the expected value.

#### Parameters

<i>expect</i>	- the expected value
<i>update</i>	- the new value

#### Returns

true if successful. False return indicates that the actual value was not equal to the expected value.

**6.122.3.2** `template<typename T> T* decaf::util::concurrent::atomic::AtomicReference< T>::get ( ) const [inline]`

Gets the Current Value.

#### Returns

the current value of this Reference.

**6.122.3.3** `template<typename T> T* decaf::util::concurrent::atomic::AtomicReference< T>::getAndSet ( T * newValue ) [inline]`

Atomically sets to the given value and returns the old value.

#### Parameters

<i>newValue</i> -	the new value
-------------------	---------------

#### Returns

the previous value.

**6.122.3.4** `template<typename T> void decaf::util::concurrent::atomic::AtomicReference< T>::set ( T * newValue ) [inline]`

Sets the Current value of this Reference.

#### Parameters

<i>newValue</i>	The new Value of this Reference.
-----------------	----------------------------------

**6.122.3.5** `template<typename T > std::string decaf::util::concurrent::atomic::AtomicReference< T >::toString ( ) const [inline]`

Returns the String representation of the current value.

#### Returns

string representation of the current value.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/atomic/AtomicReference.h`

## 6.123 activemq::transport::failover::BackupTransport Class Reference

```
#include <src/main/activemq/transport/failover/BackupTransport.h>
```

Inheritance diagram for activemq::transport::failover::BackupTransport:

### Public Member Functions

- **BackupTransport** (**BackupTransportPool** \*failover)
- virtual **~BackupTransport** ()
- **decaf::net::URI** **getUri** () const  
*Gets the URI assigned to this Backup.*
- void **setUri** (const **decaf::net::URI** &uri)  
*Sets the URI assigned to this **Transport** (p. 3996).*
- const **Pointer**< **Transport** > & **getTransport** ()  
*Gets the currently held transport.*
- void **setTransport** (const **Pointer**< **Transport** > &transport)  
*Sets the held transport, if not NULL then start to listen for exceptions from the held transport.*
- virtual void **onException** (const **decaf::lang::Exception** &ex)  
*Event handler for an exception from a command transport.*
- bool **isClosed** () const  
*Has the **Transport** (p. 3996) been shutdown and no longer usable.*
- void **setClosed** (bool value)  
*Sets the closed flag on this **Transport** (p. 3996).*

### 6.123.1 Constructor & Destructor Documentation

6.123.1.1 `activemq::transport::failover::BackupTransport::BackupTransport ( BackupTransportPool * failover )`

6.123.1.2 `virtual activemq::transport::failover::BackupTransport::~~BackupTransport ( )`  
[virtual]

### 6.123.2 Member Function Documentation

6.123.2.1 `const Pointer<Transport>& activemq::transport::failover::BackupTransport::getTransport ( )`  
[inline]

Gets the currently held transport.

#### Returns

pointer to the held transport or NULL if not set.

6.123.2.2 `decaf::net::URI activemq::transport::failover::BackupTransport::getUri ( ) const`  
[inline]

Gets the URI assigned to this Backup.

#### Returns

the assigned URI

6.123.2.3 `bool activemq::transport::failover::BackupTransport::isClosed ( ) const`  
[inline]

Has the **Transport** (p. 3996) been shutdown and no longer usable.

#### Returns

true if the **Transport** (p. 3996)

6.123.2.4 `virtual void activemq::transport::failover::BackupTransport::onException ( const decaf::lang::Exception & ex )` [virtual]

Event handler for an exception from a command transport.

The **BackupTransport** (p. 759) closes its internal **Transport** (p. 3996) when an exception is received and returns the URI to the pool of URIs to attempt connections to.

#### Parameters



## 6.124 activemq::transport::failover::BackupTransportPool Class Reference 761

<i>ex</i>	The exception that was passed to this listener to handle.
-----------	---

Implements **activemq::transport::TransportListener** (p. 4015).

**6.123.2.5** `void activemq::transport::failover::BackupTransport::setClosed ( bool value )`  
[inline]

Sets the closed flag on this **Transport** (p. 3996).

### Parameters

<i>value</i>	- true for closed.
--------------	--------------------

**6.123.2.6** `void activemq::transport::failover::BackupTransport::setTransport ( const Pointer< Transport > & transport )` [inline]

Sets the held transport, if not NULL then start to listen for exceptions from the held transport.

### Parameters

<i>transport</i>	The transport to hold.
------------------	------------------------

**6.123.2.7** `void activemq::transport::failover::BackupTransport::setUri ( const decaf::net::URI & uri )` [inline]

Sets the URI assigned to this **Transport** (p. 3996).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**BackupTransport.h**

## 6.124 activemq::transport::failover::BackupTransportPool Class Reference

```
#include <src/main/activemq/transport/failover/BackupTransportPool.h>
```

Inheritance diagram for **activemq::transport::failover::BackupTransportPool**:

### Public Member Functions

- **BackupTransportPool** (const **Pointer**< **CompositeTaskRunner** > &taskRun-

ner, const **Pointer**< **CloseTransportsTask** > &closeTask, const **Pointer**< **URIPool** > &uriPool)

- **BackupTransportPool** (int backupPoolSize, const **Pointer**< **CompositeTaskRunner** > &taskRunner, const **Pointer**< **CloseTransportsTask** > &closeTask, const **Pointer**< **URIPool** > &uriPool)
- virtual ~**BackupTransportPool** ()
- virtual bool **isPending** () const

*Return true if we don't currently have enough Connected Transports.*

- **Pointer**< **BackupTransport** > **getBackup** ()

*Get a Connected **Transport** (p. 3996) from the pool of Backups if any are present, otherwise it return a NULL Pointer.*

- virtual bool **iterate** ()

*Connect to a Backup Broker if we haven't already connected to the max number of Backups.*

- int **getBackupPoolSize** () const

*Gets the Max number of Backups this Task will create.*

- void **setBackupPoolSize** (int size)

*Sets the Max number of Backups this Task will create.*

- bool **isEnabled** () const

*Gets if the backup **Transport** (p. 3996) Pool has been enabled or not, when not enabled no backups are created and any that were are destroyed.*

- void **setEnabled** (bool value)

*Sets if this Backup **Transport** (p. 3996) Pool is enabled.*

## Friends

- class **BackupTransport**

### 6.124.1 Constructor & Destructor Documentation

6.124.1.1 `activemq::transport::failover::BackupTransportPool::BackupTransportPool ( const Pointer< CompositeTaskRunner > & taskRunner, const Pointer< CloseTransportsTask > & closeTask, const Pointer< URIPool > & uriPool )`

6.124.1.2 `activemq::transport::failover::BackupTransportPool::BackupTransportPool ( int backupPoolSize, const Pointer< CompositeTaskRunner > & taskRunner, const Pointer< CloseTransportsTask > & closeTask, const Pointer< URIPool > & uriPool )`

6.124.1.3 `virtual activemq::transport::failover::BackupTransportPool::~~BackupTransportPool ( ) [virtual]`

### 6.124.2 Member Function Documentation

6.124.2.1 `Pointer<BackupTransport> activemq::transport::failover::BackupTransportPool::getBackup ( )`

Get a Connected **Transport** (p. 3996) from the pool of Backups if any are present, otherwise it return a NULL Pointer.

#### Returns

Pointer to a Connected **Transport** (p. 3996) or NULL

6.124.2.2 `int activemq::transport::failover::BackupTransportPool::getBackupPoolSize ( ) const [inline]`

Gets the Max number of Backups this Task will create.

#### Returns

the max number of active BackupTransports that will be created.

6.124.2.3 `bool activemq::transport::failover::BackupTransportPool::isEnabled ( ) const [inline]`

Gets if the backup **Transport** (p. 3996) Pool has been enabled or not, when not enabled no backups are created and any that were are destroyed.

#### Returns

true if enable.

**6.124.2.4** `virtual bool activemq::transport::failover::BackupTransportPool::isPending ( ) const`  
`[virtual]`

Return true if we don't currently have enough Connected Transports.

Implements **activemq::threads::CompositeTask** (p. 1255).

**6.124.2.5** `virtual bool activemq::transport::failover::BackupTransportPool::iterate ( )`  
`[virtual]`

Connect to a Backup Broker if we haven't already connected to the max number of Backups.

Implements **activemq::threads::Task** (p. 3847).

**6.124.2.6** `void activemq::transport::failover::BackupTransportPool::setBackupPoolSize ( int size`  
`) [inline]`

Sets the Max number of Backups this Task will create.

#### Parameters

<i>size</i>	- the max number of active BackupTransports that will be created.
-------------	---

**6.124.2.7** `void activemq::transport::failover::BackupTransportPool::setEnabled ( bool value )`

Sets if this Backup **Transport** (p. 3996) Pool is enabled.

When not enabled no Backups are created and any that were are destroyed.

#### Parameters

<i>value</i>	- true to enable backup creation, false to disable.
--------------	---

### 6.124.3 Friends And Related Function Documentation

**6.124.3.1** `friend class BackupTransport` `[friend]`

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/BackupTransportPool.h`

## 6.125 activemq::commands::BaseCommand Class Reference

```
#include <src/main/activemq/commands/BaseCommand.h>
```

Inheritance diagram for activemq::commands::BaseCommand:

## Public Member Functions

- **BaseCommand** ()
- virtual **~BaseCommand** ()
- virtual void **setCommandId** (int id)  
*Sets the **Command** (p. 1227) Id of this **Message** (p. 2596).*
- virtual int **getCommandId** () const  
*Gets the **Command** (p. 1227) Id of this **Message** (p. 2596).*
- virtual void **setResponseRequired** (const bool required)  
*Set if this **Message** (p. 2596) requires a **Response** (p. 3379).*
- virtual bool **isResponseRequired** () const  
*Is a **Response** (p. 3379) required for this **Command** (p. 1227).*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual bool **isConnectionInfo** () const
- virtual bool **isConsumerInfo** () const
- virtual bool **isBrokerInfo** () const
- virtual bool **isMessage** () const
- virtual bool **isMessageAck** () const
- virtual bool **isKeepAliveInfo** () const
- virtual bool **isMessageDispatch** () const
- virtual bool **isMessageDispatchNotification** () const
- virtual bool **isProducerAck** () const
- virtual bool **isProducerInfo** () const
- virtual bool **isResponse** () const
- virtual bool **isRemoveInfo** () const
- virtual bool **isRemoveSubscriptionInfo** () const
- virtual bool **isShutdownInfo** () const
- virtual bool **isTransactionInfo** () const
- virtual bool **isWireFormatInfo** () const

### 6.125.1 Constructor & Destructor Documentation

6.125.1.1 `activemq::commands::BaseCommand::BaseCommand ( )` [inline]

6.125.1.2 `virtual activemq::commands::BaseCommand::~~BaseCommand ( )` [inline, virtual]

### 6.125.2 Member Function Documentation

6.125.2.1 `virtual void activemq::commands::BaseCommand::copyDataStructure ( const DataStructure * src )` [inline, virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

`src` - Source Object

Implements `activemq::commands::DataStructure` (p. 1715).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 186), `activemq::commands::ActiveMQMapMessage` (p. 219), `activemq::commands::ActiveMQMapMessage` (p. 354), `activemq::commands::ActiveMQMessage` (p. 391), `activemq::commands::ActiveMQObjectMessage` (p. 439), `activemq::commands::ActiveMQStreamMessage` (p. 540), `activemq::commands::ActiveMQTextMessage` (p. 669), `activemq::commands::BrokerError` (p. 870), `activemq::commands::BrokerInfo` (p. 904), `activemq::commands::ConnectionControl` (p. 1303), `activemq::commands::ConnectionError` (p. 1334), `activemq::commands::ConnectionInfo` (p. 1395), `activemq::commands::ConsumerControl` (p. 1443), `activemq::commands::ConsumerInfo` (p. 1504), `activemq::commands::ControlCommand` (p. 1537), `activemq::commands::DataArrayResponse` (p. 1573), `activemq::commands::DataResponse` (p. 1633), `activemq::commands::DestinationInfo` (p. 1781), `activemq::commands::ExceptionResponse` (p. 1896), `activemq::commands::FlushCommand` (p. 2000), `activemq::commands::IntegerResponse` (p. 2161), `activemq::commands::KeepAliveInfo` (p. 2336), `activemq::commands::Message` (p. 2602), `activemq::commands::MessageAck` (p. 2645), `activemq::commands::MessageDispatch` (p. 2680), `activemq::commands::MessageDispatchNotification` (p. 2718), `activemq::commands::MessagePull` (p. 2826), `activemq::commands::ProducerAck` (p. 3126), `activemq::commands::ProducerInfo` (p. 3187), `activemq::commands::RemoveInfo` (p. 3285), `activemq::commands::RemoveSubscriptionInfo` (p. 3315), `activemq::commands::ReplayCommand` (p. 3345), `activemq::commands::Response` (p. 3380), `activemq::commands::SessionInfo` (p. 3506), `activemq::commands::ShutdownInfo` (p. 3574), `activemq::commands::TransactionInfo` (p. 3961), and `activemq::commands::WireFormatInfo` (p. 4097).

References `getCommandId()`, and `isResponseRequired()`.

6.125.2.2 `virtual bool activemq::commands::BaseCommand::equals ( const DataStructure * value ) const` [inline, virtual]

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns**

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1716).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 187), **activemq::commands::ActiveMQBytesMessage** (p. 220), **activemq::commands::ActiveMQMapMessage** (p. 355), **activemq::commands::ActiveMQMessage** (p. 392), **activemq::commands::ActiveMQMessageTemplate< T >** (p. 423), **activemq::commands::ActiveMQObjectMessage** (p. 439), **activemq::commands::ActiveMQStreamMessage** (p. 540), **activemq::commands::ActiveMQTextMessage** (p. 669), **activemq::commands::BrokerInfo** (p. 905), **activemq::commands::ConnectionControl** (p. 1303), **activemq::commands::ConnectionError** (p. 1334), **activemq::commands::ConnectionInfo** (p. 1395), **activemq::commands::ConsumerControl** (p. 1443), **activemq::commands::ConsumerInfo** (p. 1504), **activemq::commands::ControlCommand** (p. 1538), **activemq::commands::DataArrayResponse** (p. 1573), **activemq::commands::DataResponse** (p. 1633), **activemq::commands::DestinationInfo** (p. 1781), **activemq::commands::ExceptionResponse** (p. 1896), **activemq::commands::FlushCommand** (p. 2000), **activemq::commands::IntegerResponse** (p. 2161), **activemq::commands::KeepAliveInfo** (p. 2337), **activemq::commands::Message** (p. 2602), **activemq::commands::MessageAck** (p. 2645), **activemq::commands::MessageDispatch** (p. 2680), **activemq::commands::MessageDispatchNotification** (p. 2718), **activemq::commands::MessagePull** (p. 2826), **activemq::commands::ProducerAck** (p. 3126), **activemq::commands::ProducerInfo** (p. 3187), **activemq::commands::RemoveInfo** (p. 3286), **activemq::commands::RemoveSubscriptionInfo** (p. 3315), **activemq::commands::ReplayCommand** (p. 3345), **activemq::commands::Response** (p. 3381), **activemq::commands::SessionInfo** (p. 3507), **activemq::commands::ShutdownInfo** (p. 3574), **activemq::commands::TransactionInfo** (p. 3961), **activemq::commands::WireFormatInfo** (p. 4097), **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 423), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 423), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 423), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 423), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 423), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 423).

References **activemq::commands::BaseDataStructure::equals()**.

**6.125.2.3** `virtual int activemq::commands::BaseCommand::getCommandId ( ) const`  
`[inline, virtual]`

Gets the **Command** (p. 1227) Id of this **Message** (p. 2596).

**Returns**

**Command** (p. 1227) Id

Implements **activemq::commands::Command** (p. 1228).

Referenced by **copyDataStructure()**.

**6.125.2.4** `virtual bool activemq::commands::BaseCommand::isBrokerInfo ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1228).

Reimplemented in **activemq::commands::BrokerInfo** (p. 907).

**6.125.2.5** `virtual bool activemq::commands::BaseCommand::isConnectionInfo ( ) const`  
[inline, virtual]

Implements **activemq::commands::Command** (p. 1229).

Reimplemented in **activemq::commands::ConnectionInfo** (p. 1397).

**6.125.2.6** `virtual bool activemq::commands::BaseCommand::isConsumerInfo ( ) const`  
[inline, virtual]

Implements **activemq::commands::Command** (p. 1229).

Reimplemented in **activemq::commands::ConsumerInfo** (p. 1506).

**6.125.2.7** `virtual bool activemq::commands::BaseCommand::isKeepAliveInfo ( ) const`  
[inline, virtual]

Implements **activemq::commands::Command** (p. 1229).

Reimplemented in **activemq::commands::KeepAliveInfo** (p. 2337).

**6.125.2.8** `virtual bool activemq::commands::BaseCommand::isMessage ( ) const`  
[inline, virtual]

Implements **activemq::commands::Command** (p. 1229).

Reimplemented in **activemq::commands::Message** (p. 2608).

**6.125.2.9** `virtual bool activemq::commands::BaseCommand::isMessageAck ( ) const`  
[inline, virtual]

Implements **activemq::commands::Command** (p. 1229).

Reimplemented in **activemq::commands::MessageAck** (p. 2646).

**6.125.2.10** `virtual bool activemq::commands::BaseCommand::isMessageDispatch ( ) const`  
[inline, virtual]

Implements **activemq::commands::Command** (p. 1229).

Reimplemented in **activemq::commands::MessageDispatch** (p. 2681).

**6.125.2.11** `virtual bool activemq::commands::BaseCommand::isMessageDispatchNotification ( ) const` [inline, virtual]

Implements **activemq::commands::Command** (p. 1229).



Reimplemented in **activemq::commands::MessageDispatchNotification** (p. 2719).

**6.125.2.12** `virtual bool activemq::commands::BaseCommand::isProducerAck ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1230).

Reimplemented in **activemq::commands::ProducerAck** (p. 3127).

**6.125.2.13** `virtual bool activemq::commands::BaseCommand::isProducerInfo ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1230).

Reimplemented in **activemq::commands::ProducerInfo** (p. 3188).

**6.125.2.14** `virtual bool activemq::commands::BaseCommand::isRemoveInfo ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1230).

Reimplemented in **activemq::commands::RemoveInfo** (p. 3286).

**6.125.2.15** `virtual bool activemq::commands::BaseCommand::isRemoveSubscriptionInfo ( )`  
`const [inline, virtual]`

Implements **activemq::commands::Command** (p. 1230).

Reimplemented in **activemq::commands::RemoveSubscriptionInfo** (p. 3316).

**6.125.2.16** `virtual bool activemq::commands::BaseCommand::isResponse ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1230).

Reimplemented in **activemq::commands::Response** (p. 3381).

**6.125.2.17** `virtual bool activemq::commands::BaseCommand::isResponseRequired ( ) const`  
`[inline, virtual]`

Is a **Response** (p. 3379) required for this **Command** (p. 1227).

#### Returns

true if a response is required.

Implements **activemq::commands::Command** (p. 1230).

Referenced by `copyDataStructure()`.

**6.125.2.18** `virtual bool activemq::commands::BaseCommand::isShutdownInfo ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1230).

Reimplemented in **activemq::commands::ShutdownInfo** (p. 3575).

**6.125.2.19** `virtual bool activemq::commands::BaseCommand::isTransactionInfo ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1231).

Reimplemented in **activemq::commands::TransactionInfo** (p. 3962).

**6.125.2.20** `virtual bool activemq::commands::BaseCommand::isWireFormatInfo ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1231).

Reimplemented in **activemq::commands::WireFormatInfo** (p. 4101).

**6.125.2.21** `virtual void activemq::commands::BaseCommand::setCommandId ( int id )`  
`[inline, virtual]`

Sets the **Command** (p. 1227) Id of this **Message** (p. 2596).

#### Parameters

<i>id</i>	<b>Command</b> (p. 1227) Id
-----------	-----------------------------

Implements **activemq::commands::Command** (p. 1231).

**6.125.2.22** `virtual void activemq::commands::BaseCommand::setResponseRequired ( const bool required )` `[inline, virtual]`

Set if this **Message** (p. 2596) requires a **Response** (p. 3379).

#### Parameters

<i>required</i>	true if response is required
-----------------	------------------------------

Implements **activemq::commands::Command** (p. 1231).

**6.125.2.23** `virtual std::string activemq::commands::BaseCommand::toString ( ) const`  
`[inline, virtual]`

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Implements `activemq::commands::Command` (p. 1231).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 189), `activemq::commands::ActiveMQBytesMessage` (p. 229), `activemq::commands::ActiveMQMapMessage` (p. 364), `activemq::commands::ActiveMQMessage` (p. 392), `activemq::commands::ActiveMQObjectMessage` (p. 440), `activemq::commands::ActiveMQStreamMessage` (p. 547), `activemq::commands::ActiveMQTextMessage` (p. 671), `activemq::commands::BrokerInfo` (p. 908), `activemq::commands::ConnectionControl` (p. 1305), `activemq::commands::ConnectionError` (p. 1335), `activemq::commands::ConnectionInfo` (p. 1398), `activemq::commands::ConsumerControl` (p. 1445), `activemq::commands::ConsumerInfo` (p. 1507), `activemq::commands::ControlCommand` (p. 1538), `activemq::commands::DataArrayResponse` (p. 1574), `activemq::commands::DataResponse` (p. 1634), `activemq::commands::DestinationInfo` (p. 1783), `activemq::commands::ExceptionResponse` (p. 1897), `activemq::commands::FlushCommand` (p. 2001), `activemq::commands::IntegerResponse` (p. 2162), `activemq::commands::KeepAliveInfo` (p. 2337), `activemq::commands::Message` (p. 2611), `activemq::commands::MessageAck` (p. 2647), `activemq::commands::MessageDispatch` (p. 2682), `activemq::commands::MessageDispatchNotification` (p. 2719), `activemq::commands::MessagePull` (p. 2827), `activemq::commands::ProducerAck` (p. 3127), `activemq::commands::ProducerInfo` (p. 3189), `activemq::commands::RemoveInfo` (p. 3287), `activemq::commands::RemoveSubscriptionInfo` (p. 3317), `activemq::commands::ReplayCommand` (p. 3346), `activemq::commands::Response` (p. 3382), `activemq::commands::SessionInfo` (p. 3507), `activemq::commands::ShutdownInfo` (p. 3575), `activemq::commands::TransactionInfo` (p. 3963), and `activemq::commands::WireFormatInfo` (p. 4104).

References `activemq::commands::BaseDataStructure::toString()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BaseCommand.h`

## 6.126 `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 771).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller`:

### Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.126.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 771). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.126.2 Constructor & Destructor Documentation

6.126.2.1 **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::BaseCommandMarshaller** ( ) [inline]

6.126.2.2 **virtual activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::~~BaseCommandMarshaller** ( ) [inline, virtual]

### 6.126.3 Member Function Documentation

6.126.3.1 **virtual void activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::looseMarshal** ( **OpenWireFormat** \* wireFormat, **commands::DataStructure** \* dataStructure, **decaf::io::DataOutputStream** \* dataOut ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 191), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 236), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 366), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 394), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 442), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 555), **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 673), **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller** (p. 912), **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller** (p. 1308), **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller** (p. 1342), **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller** (p. 1406), **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller** (p. 1453), **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller** (p. 1516), **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller** (p. 1545), **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1581), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1649), **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller** (p. 1790), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1908), **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller** (p. 2012), **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 2173), **activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller** (p. 2348), **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller** (p. 2659), **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller** (p. 2696), **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller** (p. 2731), **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2786), **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller** (p. 2839), **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller** (p. 3139), **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller** (p. 3209), **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller** (p. 3298), **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller** (p. 3324), **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller** (p. 3361), **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3405), **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller** (p. 3523), **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller** (p. 3595), and **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller** (p. 3974).

```
6.126.3.2 virtual void activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1683).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 192), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 237), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 367), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 395), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 443), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 555), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 674), `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 913), `activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1309), `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1342), `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1406), `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1453), `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1516), `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1546), `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1581), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1650), `activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1791), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1908), `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 2012), `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2173), `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 2349), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller` (p. 2659), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller` (p. 2697), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller` (p. 2731), `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2786), `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2839), `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 3139), `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 3209), `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3298), `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller` (p. 3324), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller`

## 6.126 activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller

---

Class Reference775

(p. 3362), [activemq::wireformat::openwire::marshal::v3::ResponseMarshaller](#) (p. 3406), [activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller](#) (p. 3523), [activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller](#) (p. 3595), and [activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller](#) (p. 3975).

**6.126.3.3** `virtual int activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements [activemq::wireformat::openwire::marshal::DataStreamMarshaller](#) (p. 1690).

Reimplemented in [activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller](#) (p. 192), [activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller](#) (p. 237), [activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller](#) (p. 367), [activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller](#) (p. 395), [activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller](#) (p. 443), [activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller](#) (p. 556), [activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller](#) (p. 674), [activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller](#) (p. 913), [activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller](#) (p. 1309), [activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller](#) (p. 1343), [activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller](#) (p. 1407), [activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller](#) (p. 1454), [activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller](#) (p. 1517), [activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller](#) (p. 1546), [activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller](#) (p. 1581), [activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller](#) (p. 1650), [activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller](#) (p. 1791), [activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller](#) (p. 1909), [activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller](#) (p. 2013), [activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller](#) (p. 2174), [activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller](#) (p. 2349),

`activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller` (p. 2660),  
`activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller` (p. 2697),  
`activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller`  
(p. 2732), `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2787),  
`activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2840),  
`activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 3140),  
`activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 3210),  
`activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3299),  
`activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller`  
(p. 3325), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller`  
(p. 3362), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3407),  
`activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 3524),  
`activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3596),  
and `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3975).

```
6.126.3.4 virtual void activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1698).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller`  
(p. 193), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller`  
(p. 238), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller`  
(p. 368), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller`  
(p. 396), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller`  
(p. 444), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller`  
(p. 556), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller`  
(p. 675), `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 914),  
`activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1310),  
`activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1343),  
`activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1407),  
`activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1454),  
`activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1517),



**activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller** (p. 1547),  
**activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1582),  
**activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1651),  
**activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller** (p. 1792),  
**activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1909),  
**activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller** (p. 2013),  
**activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 2174),  
**activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller** (p. 2350),  
**activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller** (p. 2660),  
**activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller** (p. 2698),  
**activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller**  
(p. 2732), **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2788),  
**activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller** (p. 2840),  
**activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller** (p. 3140),  
**activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller** (p. 3210),  
**activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller** (p. 3299),  
**activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller**  
(p. 3325), **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller**  
(p. 3363), **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3407),  
**activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller** (p. 3524),  
**activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller** (p. 3596),  
and **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller** (p. 3976).

**6.126.3.5** virtual void **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::tightUnmarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream**  
\* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller**  
(p. 193), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller**  
(p. 238), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller**  
(p. 368), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller**  
(p. 396), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller**  
(p. 444), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller**

(p. 557), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 675), `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 914), `activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1310), `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1344), `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1408), `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1455), `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1518), `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1547), `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1582), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1651), `activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1792), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1910), `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 2014), `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2175), `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 2350), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller` (p. 2661), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller` (p. 2698), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller` (p. 2733), `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2788), `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2841), `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 3141), `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 3211), `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3300), `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller` (p. 3326), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller` (p. 3363), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3408), `activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 3525), `activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3597), and `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3976).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h`

## 6.127 `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for `BaseCommandMarshaller` (p. 778).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller`:

### Public Member Functions

- `BaseCommandMarshaller()`

- virtual ~**BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.127.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 778). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.127.2 Constructor & Destructor Documentation

- 6.127.2.1 `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::BaseCommandMarshaller ( ) [inline]`
- 6.127.2.2 `virtual activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::~~BaseCommandMarshaller ( ) [inline, virtual]`

## 6.127.3 Member Function Documentation

- 6.127.3.1 `virtual void activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1675).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 200), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 245), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 375), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 403), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 451), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 563), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 677), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 916), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1313), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1346), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1410), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1457), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1520), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1550), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1585), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1654), `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1795), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1917), `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 2016), `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2177),

**activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller** (p. 2353),  
**activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller** (p. 2663),  
**activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller** (p. 2705),  
**activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller**  
(p. 2735), **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2795),  
**activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller** (p. 2843),  
**activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller** (p. 3134),  
**activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller** (p. 3192),  
**activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller** (p. 3311),  
**activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller**  
(p. 3341), **activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller**  
(p. 3349), **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3390),  
**activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller** (p. 3531),  
**activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller** (p. 3599),  
and **activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller** (p. 3983).

**6.127.3.2** virtual void **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::looseUnmarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
\* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller**  
(p. 200), **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller**  
(p. 245), **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller**  
(p. 375), **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller**  
(p. 403), **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller**  
(p. 451), **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller**  
(p. 564), **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller**  
(p. 678), **activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller** (p. 917),  
**activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller** (p. 1313),  
**activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller** (p. 1347),  
**activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller** (p. 1410),  
**activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller** (p. 1457),  
**activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller** (p. 1520),  
**activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller** (p. 1550),

**activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1585),  
**activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1654),  
**activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller** (p. 1795),  
**activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1917),  
**activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller** (p. 2017),  
**activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 2178),  
**activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller** (p. 2353),  
**activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller** (p. 2663),  
**activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller** (p. 2705),  
**activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller**  
 (p. 2735), **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2795),  
**activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller** (p. 2843),  
**activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller** (p. 3135),  
**activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller** (p. 3192),  
**activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller** (p. 3311),  
**activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller**  
 (p. 3341), **activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller**  
 (p. 3349), **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3391),  
**activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller** (p. 3532),  
**activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller** (p. 3599),  
 and **activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller** (p. 3983).

**6.127.3.3** `virtual int activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::tightMarshal1`  
`( OpenWireFormat * wireFormat, commands::DataStructure *`   
`dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
`[virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller**  
 (p. 201), **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller**  
 (p. 246), **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller**  
 (p. 376), **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller**

(p. 404), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 452), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 564), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 678), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 917), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1314), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1347), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1411), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1458), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1521), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1551), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1586), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1654), `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1796), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1917), `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 2017), `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2178), `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2354), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2664), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2706), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller` (p. 2736), `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2796), `activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2844), `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 3135), `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 3193), `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 3312), `activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller` (p. 3342), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller` (p. 3350), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3392), `activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 3532), `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 3600), and `activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3984).

```
6.127.3.4 virtual void activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the marshal.
---

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1698).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 201), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 246), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 376), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 404), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 452), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 565), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 679), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 918), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1314), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1348), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1411), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1458), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1521), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1551), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1586), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1655), `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1796), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1918), `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 2018), `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2179), `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2354), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2664), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2706), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller` (p. 2736), `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2797), `activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2844), `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 3136), `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 3193), `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 3312), `activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller` (p. 3342), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller` (p. 3350), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3392), `activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 3533), `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 3600), and `activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3984).

```
6.127.3.5  virtual void activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
  * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.



### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1706).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 202), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 247), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 377), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 405), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 453), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 565), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 679), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 918), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1315), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1348), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1412), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1459), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1522), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1552), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1587), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1655), `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1797), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1918), `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 2018), `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2179), `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2355), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2665), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2707), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller` (p. 2737), `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2797), `activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2845), `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 3136), `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 3194), `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 3313), `activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller` (p. 3343), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller` (p. 3351), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3393), `activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 3533), `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 3601), and `activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3985).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h`

## 6.128 `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 786).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller`:

### Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

## 6.128.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 786). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.128.2 Constructor & Destructor Documentation

6.128.2.1 **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::BaseCommandMarshaller**  
( ) [inline]

6.128.2.2 **virtual activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::~BaseCommandMarshaller**  
( ) [inline, virtual]

## 6.128.3 Member Function Documentation

6.128.3.1 **virtual void activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::looseMarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
\* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 195), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 241), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 370), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 399), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 446), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 559), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 682), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 921), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 1317), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 1350), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1414), **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller** (p. 1461), **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller** (p. 1524), **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller** (p. 1554),

**activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1589),  
**activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1658),  
**activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller** (p. 1799),  
**activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1921),  
**activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller** (p. 2020),  
**activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 2182),  
**activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller** (p. 2361),  
**activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller** (p. 2667),  
**activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller** (p. 2709),  
**activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller**  
 (p. 2739), **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2799),  
**activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller** (p. 2847),  
**activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller** (p. 3151),  
**activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller** (p. 3200),  
**activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller** (p. 3302),  
**activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller**  
 (p. 3320), **activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller**  
 (p. 3353), **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3410),  
**activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller** (p. 3519),  
**activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller** (p. 3586),  
 and **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller** (p. 3970).

**6.128.3.2** `virtual void activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::looseUnmarshal  
 ( OpenWireFormat * wireFormat, commands::DataStructure  
 * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller**  
 (p. 196), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller**  
 (p. 241), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller**  
 (p. 371), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller**  
 (p. 399), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller**  
 (p. 447), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller**  
 (p. 559), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller**  
 (p. 682), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 921),

activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (p. 1317),  
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 1351),  
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1415),  
activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1462),  
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1525),  
activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1554),  
activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1590),  
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1658),  
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1799),  
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1921),  
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 2021),  
activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 2182),  
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 2362),  
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (p. 2668),  
activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (p. 2710),  
activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller  
(p. 2740), activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2800),  
activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2848),  
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 3152),  
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 3201),  
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 3303),  
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller  
(p. 3320), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller  
(p. 3353), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 3411),  
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 3519),  
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 3587),  
and activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3970).

```

6.128.3.3  virtual int activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1690).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 196), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 242), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 371), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 400), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 447), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 560), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 683), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 922), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1318), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1351), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1415), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1462), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1525), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1555), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1590), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1659), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1800), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1922), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 2021), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2182), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2362), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2668), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2710), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller` (p. 2740), `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2800), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2848), `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 3152), `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 3201), `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3303), `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller` (p. 3321), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller` (p. 3354), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3412), `activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 3520), `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3587), and `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3971).

```
6.128.3.4  virtual void activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure
              * dataStructure, decaf::io::DataOutputStream * dataOut,
              utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 197), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 242), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 372), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 400), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 448), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 560), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 683), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 922), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 1318), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 1352), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1416), **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller** (p. 1463), **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller** (p. 1526), **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller** (p. 1555), **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1591), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1659), **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller** (p. 1800), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1922), **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller** (p. 2022), **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 2183), **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller** (p. 2363), **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller** (p. 2669), **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller** (p. 2711), **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller** (p. 2741), **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2801), **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller** (p. 2849), **activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller** (p. 3153), **activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller** (p. 3202), **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller** (p. 3304), **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller** (p. 3321), **activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller** (p. 3354), **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3412), **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller** (p. 3520), **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller** (p. 3588), and **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller** (p. 3971).

```
6.128.3.5 virtual void activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1706).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 197), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 243), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 372), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 401), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 448), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 561), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 684), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 923), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1319), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1352), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1416), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1463), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1526), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1556), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1591), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1660), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1801), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1923), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 2022), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2183), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2363), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2669), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2711), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller` (p. 2741), `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2802), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2849), `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 3153), `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 3202), `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3304), `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller`



## 6.129 activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller Class Reference 793

(p. 3322), **activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller** (p. 3355), **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3413), **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller** (p. 3521), **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller** (p. 3588), and **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller** (p. 3972).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h`

## 6.129 activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 793).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller`:

### Public Member Functions

- **BaseCommandMarshaller ()**
- virtual **~BaseCommandMarshaller ()**
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)**  
**throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**

*Write a object instance to data output stream.*

### 6.129.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 793). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.129.2 Constructor & Destructor Documentation

6.129.2.1 **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::BaseCommandMarshaller**  
( ) [inline]

6.129.2.2 **virtual activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::~~BaseCommandMarshaller**  
( ) [inline, virtual]

### 6.129.3 Member Function Documentation

6.129.3.1 **virtual void activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::looseMarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
\* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw (  
**decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 208), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 249), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 379), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 407), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 455), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 567), **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 686), **activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller** (p. 925), **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller** (p. 1321), **activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller** (p. 1355),

`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1418),  
`activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1465),  
`activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1528),  
`activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1558),  
`activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1594),  
`activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1636),  
`activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1807),  
`activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1912),  
`activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 2025),  
`activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2186),  
`activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 2357),  
`activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller` (p. 2671),  
`activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller` (p. 2701),  
`activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller`  
(p. 2743), `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2781),  
`activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2834),  
`activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 3143),  
`activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 3205),  
`activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3306),  
`activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller`  
(p. 3337), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller`  
(p. 3370), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3400),  
`activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 3514),  
`activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 3590),  
and `activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3966).

**6.129.3.2** `virtual void activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::looseUnmarshal`  
( `OpenWireFormat * wireFormat`, `commands::DataStructure`  
\* `dataStructure`, `decaf::io::DataInputStream * dataIn` ) throw ( `decaf::io::IOException` ) `[virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- <code>BinaryReader</code> that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1683).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller`  
(p. 209), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`  
(p. 250), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller`  
(p. 379), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller`

(p. 408), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 455), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 568), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 686), `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 925), `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1322), `activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1355), `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1419), `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1466), `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1529), `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1559), `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1594), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1637), `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1808), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1913), `activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 2025), `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2186), `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 2357), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller` (p. 2672), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller` (p. 2701), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller` (p. 2744), `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2782), `activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2835), `activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 3143), `activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 3205), `activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3307), `activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller` (p. 3337), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller` (p. 3370), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3401), `activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 3515), `activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 3591), and `activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3966).

```
6.129.3.3  virtual int activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 209), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 250), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 380), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 408), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 456), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 568), **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 687), **activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller** (p. 926), **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller** (p. 1322), **activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller** (p. 1356), **activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller** (p. 1419), **activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller** (p. 1466), **activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller** (p. 1529), **activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller** (p. 1559), **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller** (p. 1594), **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller** (p. 1637), **activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller** (p. 1808), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1913), **activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller** (p. 2026), **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller** (p. 2187), **activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller** (p. 2358), **activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller** (p. 2672), **activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller** (p. 2702), **activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller** (p. 2744), **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2783), **activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller** (p. 2835), **activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller** (p. 3144), **activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller** (p. 3206), **activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller** (p. 3307), **activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller** (p. 3338), **activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller** (p. 3371), **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3402), **activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller** (p. 3515), **activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller** (p. 3591), and **activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller** (p. 3967).

```
6.129.3.4 virtual void activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1698).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 210), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 251), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 380), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 409), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 456), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 569), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 687), `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 926), `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1323), `activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1356), `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1420), `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1467), `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1530), `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1560), `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1595), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1638), `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1809), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1914), `activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 2026), `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2187), `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 2358), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller` (p. 2673), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller` (p. 2702), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller` (p. 2745), `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2783), `activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2836), `activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 3144), `activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 3206), `activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3308), `activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller` (p. 3338), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller` (p. 3371), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3402), `activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 3516), `activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 3592), and `activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3967).

6.129.3.5 virtual void activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 210), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 251), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 381), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 409), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 457), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 569), **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 688), **activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller** (p. 927), **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller** (p. 1323), **activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller** (p. 1357), **activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller** (p. 1420), **activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller** (p. 1467), **activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller** (p. 1530), **activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller** (p. 1560), **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller** (p. 1595), **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller** (p. 1638), **activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller** (p. 1809), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1914), **activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller** (p. 2027), **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller** (p. 2188), **activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller** (p. 2359), **activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller** (p. 2673), **activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller** (p. 2703), **activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller** (p. 2745), **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2784), **activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller** (p. 2836), **activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller** (p. 3145), **activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller** (p. 3207), **activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller** (p. 3308), **activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller**

(p. 3339), **activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller** (p. 3372), **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3403), **activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller** (p. 3516), **activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller** (p. 3592), and **activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller** (p. 3968).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h`

### 6.130 **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 800).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller**:

#### Public Member Functions

- **BaseCommandMarshaller ()**
- virtual **~BaseCommandMarshaller ()**
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)**  
throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn)** throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut)** throw ( decaf::io::IOException )



*Write a object instance to data output stream.*

### 6.130.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 800). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.130.2 Constructor & Destructor Documentation

**6.130.2.1** `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::BaseCommandMarshaller ( ) [inline]`

**6.130.2.2** `virtual activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::~~BaseCommandMarshaller ( ) [inline, virtual]`

### 6.130.3 Member Function Documentation

**6.130.3.1** `virtual void activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

*Write a object instance to data output stream.*

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller** (p. 212), **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller** (p. 253), **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller** (p. 387), **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller** (p. 416), **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller** (p. 463), **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller** (p. 576), **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller** (p. 690), **activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller** (p. 929), **activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller** (p. 1325), **activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller** (p. 1359),

**activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller** (p. 1423),  
**activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller** (p. 1470),  
**activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller** (p. 1533),  
**activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller** (p. 1562),  
**activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller** (p. 1598),  
**activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller** (p. 1641),  
**activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller** (p. 1803),  
**activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller** (p. 1899),  
**activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller** (p. 2003),  
**activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller** (p. 2164),  
**activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller** (p. 2340),  
**activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller** (p. 2650),  
**activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller** (p. 2713),  
**activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller**  
 (p. 2722), **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2804),  
**activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller** (p. 2851),  
**activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller** (p. 3147),  
**activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller** (p. 3213),  
**activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller** (p. 3294),  
**activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller**  
 (p. 3332), **activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller**  
 (p. 3366), **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3415),  
**activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller** (p. 3510),  
**activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller** (p. 3578),  
 and **activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller** (p. 3978).

**6.130.3.2** `virtual void activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::looseUnmarshal  
 ( OpenWireFormat * wireFormat, commands::DataStructure  
 * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller**  
 (p. 213), **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller**  
 (p. 254), **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller**  
 (p. 388), **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller**

(p. 416), **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller** (p. 464), **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller** (p. 576), **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller** (p. 691), **activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller** (p. 930), **activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller** (p. 1326), **activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller** (p. 1359), **activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller** (p. 1423), **activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller** (p. 1470), **activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller** (p. 1533), **activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller** (p. 1563), **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller** (p. 1598), **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller** (p. 1641), **activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller** (p. 1804), **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller** (p. 1900), **activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller** (p. 2004), **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller** (p. 2165), **activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller** (p. 2340), **activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller** (p. 2651), **activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller** (p. 2714), **activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller** (p. 2723), **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2804), **activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller** (p. 2852), **activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller** (p. 3148), **activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller** (p. 3214), **activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller** (p. 3294), **activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller** (p. 3333), **activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller** (p. 3366), **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3416), **activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller** (p. 3511), **activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller** (p. 3578), and **activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller** (p. 3979).

**6.130.3.3** `virtual int activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i> if an error occurs during the marshal.
---

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1690).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 213), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 254), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 388), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 417), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 464), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 577), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 691), `activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 930), `activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1326), `activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1360), `activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1424), `activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1471), `activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1534), `activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1563), `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1599), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1642), `activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1804), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1900), `activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 2004), `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2165), `activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2341), `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller` (p. 2651), `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller` (p. 2714), `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller` (p. 2723), `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2805), `activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2852), `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 3148), `activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 3214), `activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 3295), `activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller` (p. 3333), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller` (p. 3367), `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3417), `activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller` (p. 3511), `activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller` (p. 3579), and `activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3979).

```
6.130.3.4  virtual void activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller** (p. 214), **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller** (p. 255), **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller** (p. 389), **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller** (p. 417), **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller** (p. 465), **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller** (p. 577), **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller** (p. 692), **activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller** (p. 931), **activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller** (p. 1327), **activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller** (p. 1360), **activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller** (p. 1424), **activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller** (p. 1471), **activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller** (p. 1534), **activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller** (p. 1564), **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller** (p. 1599), **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller** (p. 1642), **activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller** (p. 1805), **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller** (p. 1901), **activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller** (p. 2005), **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller** (p. 2166), **activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller** (p. 2341), **activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller** (p. 2652), **activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller** (p. 2715), **activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller** (p. 2724), **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2806), **activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller** (p. 2853), **activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller** (p. 3149), **activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller** (p. 3215), **activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller** (p. 3295), **activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller** (p. 3334), **activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller** (p. 3367), **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3417), **activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller** (p. 3512), **activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller** (p. 3579), and **activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller** (p. 3980).

```

6.130.3.5 virtual void activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
  * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1706).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 214), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 255), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 389), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 418), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 465), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 578), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 692), `activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 931), `activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1327), `activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1361), `activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1425), `activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1472), `activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1535), `activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1564), `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1600), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1643), `activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1805), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1901), `activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 2005), `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2166), `activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2342), `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller` (p. 2652), `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller` (p. 2715), `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller` (p. 2724), `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2806), `activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2853), `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 3149), `activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 3215), `activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 3296), `activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller`

## 6.131 activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller

### Class Reference

807

(p. 3334), **activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller** (p. 3368), **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3418), **activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller** (p. 3512), **activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller** (p. 3580), and **activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller** (p. 3980).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h`

## 6.131 activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller

### Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 807).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller`:

#### Public Member Functions

- **BaseCommandMarshaller ()**
- virtual **~BaseCommandMarshaller ()**
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)**  
**throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** **throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs)** **throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn)** **throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut)** **throw ( decaf::io::IOException )**

*Write a object instance to data output stream.*

### 6.131.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 807). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.131.2 Constructor & Destructor Documentation

6.131.2.1 **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::BaseCommandMarshaller**  
( ) [inline]

6.131.2.2 **virtual activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::~~BaseCommandMarshaller**  
( ) [inline, virtual]

### 6.131.3 Member Function Documentation

6.131.3.1 **virtual void activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::looseMarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
\* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw (  
**decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 204), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 258), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 383), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 411), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 459), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 572), **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 694), **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller** (p. 933), **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller** (p. 1330), **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller** (p. 1338),



activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1401),  
 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1448),  
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1511),  
 activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1541),  
 activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1576),  
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1645),  
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1786),  
 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1904),  
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 2008),  
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 2169),  
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 2344),  
 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 2654),  
 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (p. 2692),  
 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller  
 (p. 2726), activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2790),  
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2830),  
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 3130),  
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 3196),  
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 3289),  
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller  
 (p. 3328), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller  
 (p. 3357), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 3395),  
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 3527),  
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 3582),  
 and activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3987).

6.131.3.2 virtual void activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::looseUnmarshal  
 ( OpenWireFormat \* wireFormat, commands::DataStructure  
 \* dataStructure, decaf::io::DataInputStream \* dataIn ) throw (   
 decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller**  
 (p. 204), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller**  
 (p. 258), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller**  
 (p. 384), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller**

(p. 412), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 460), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 572), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 695), `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 934), `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1330), `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1338), `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1402), `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1449), `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1512), `activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1542), `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1577), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1645), `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1787), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1904), `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 2008), `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2169), `activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 2345), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller` (p. 2655), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller` (p. 2693), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller` (p. 2727), `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2791), `activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2831), `activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 3131), `activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 3197), `activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 3290), `activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller` (p. 3329), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller` (p. 3358), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3396), `activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 3528), `activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 3582), and `activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3987).

```
6.131.3.3  virtual int activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 205), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 259), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 384), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 412), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 460), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 573), **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 695), **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller** (p. 934), **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller** (p. 1331), **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller** (p. 1339), **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller** (p. 1402), **activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller** (p. 1449), **activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller** (p. 1512), **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller** (p. 1542), **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1577), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1646), **activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller** (p. 1787), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1905), **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller** (p. 2009), **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 2169), **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller** (p. 2345), **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller** (p. 2655), **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller** (p. 2693), **activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller** (p. 2727), **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2791), **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller** (p. 2831), **activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller** (p. 3131), **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller** (p. 3197), **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller** (p. 3290), **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller** (p. 3329), **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller** (p. 3358), **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3397), **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller** (p. 3528), **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller** (p. 3583), and **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller** (p. 3988).

```
6.131.3.4  virtual void activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1698).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 205), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 259), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 385), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 413), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 461), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 573), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 696), `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 935), `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1331), `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1339), `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1403), `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1450), `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1513), `activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1543), `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1578), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1646), `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1788), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1905), `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 2009), `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2170), `activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 2346), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller` (p. 2656), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller` (p. 2694), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller` (p. 2728), `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2792), `activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2832), `activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 3132), `activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 3198), `activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 3291), `activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller` (p. 3330), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller` (p. 3359), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3397), `activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 3529), `activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 3583), and `activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3988).

6.131.3.5 virtual void activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 206), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 260), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 385), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 413), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 461), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 574), **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 696), **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller** (p. 935), **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller** (p. 1332), **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller** (p. 1340), **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller** (p. 1403), **activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller** (p. 1450), **activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller** (p. 1513), **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller** (p. 1543), **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1578), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1647), **activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller** (p. 1788), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1906), **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller** (p. 2010), **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 2170), **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller** (p. 2346), **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller** (p. 2656), **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller** (p. 2694), **activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller** (p. 2728), **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2793), **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller** (p. 2832), **activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller** (p. 3132), **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller** (p. 3198), **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller** (p. 3291), **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller**

(p. 3330), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller` (p. 3359), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3398), `activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 3529), `activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 3584), and `activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3989).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h`

## 6.132 `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller` Class Reference

Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

Inherits `activemq::wireformat::openwire::marshal::DataStreamMarshaller`.

Inherited by `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller`, `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller`, `activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller`, `activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller`, `activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller`, `activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller`, `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller`, `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller`, `activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller`, `activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller`, `activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller`, `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller`, `activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller`, `activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller`, `activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller`, `activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller`, `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller`, `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller`, `activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller`, `activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller`, `activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller`, `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller`, `activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller`, `activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller`, `activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller`, `activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller`, `activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller`, `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller`, `activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller`, `activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller`, `activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller`.

activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller, activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller, activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller, activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller, activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller, activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller, activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller, activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller, activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller, activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller, activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller, activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller, activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller, activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller, activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller, activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller, activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller, activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller, activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller, activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller, activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller, activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller, activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller, activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller, activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller, activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller, and activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller.

## Public Member Functions

- virtual `~BaseDataStreamMarshaller()`
- virtual `int tightMarshal1 (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED) throw ( decaf::io::IOException )`  
*Tight Marshal to the given stream.*
- virtual `void tightMarshal2 (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataOutputStream *ds AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED) throw ( decaf::io::IOException )`  
*Tight Marshal to the given stream.*
- virtual `void tightUnmarshal (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataInputStream`

\*dis AMQCPP\_UNUSED, **utils::BooleanStream** \*bs AMQCPP\_UNUSED) throw  
( decaf::io::IOException )

*Tight Un-Marshall to the given stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*format AMQCPP\_UNUSED, **commands::DataStructure** \*command AMQCPP\_UNUSED, **decaf::io::DataOutputStream** \*ds AMQCPP\_UNUSED) throw ( decaf::io::IOException )

*Tight Marshal to the given stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*format AMQCPP\_UNUSED, **commands::DataStructure** \*command AMQCPP\_UNUSED, **decaf::io::DataInputStream** \*dis AMQCPP\_UNUSED) throw ( decaf::io::IOException )

*Loose Un-Marshall to the given stream.*

### Static Public Member Functions

- static std::string **toString** (const **commands::MessageId** \*id)  
*Converts the object to a String.*
- static std::string **toString** (const **commands::ProducerId** \*id)  
*Converts the object to a String.*
- static std::string **toString** (const **commands::TransactionId** \*txnId)  
*Converts the given transaction ID into a String.*
- static std::string **toHexFromBytes** (const std::vector< unsigned char > &data)  
*given an array of bytes, convert that array to a Hexidecimal coded string that represents that data.*

### Protected Member Functions

- virtual **commands::DataStructure** \* **tightUnmarshalCachedObject** (**OpenWireFormat** \*wireFormat, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Tight Unmarshal the cached object.*
- virtual int **tightMarshalCachedObject1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*data, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.*
- virtual void **tightMarshalCachedObject2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*data, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )



*Tightly marshals the passed DataStructure based object to the passed streams returning nothing.*

- virtual void **looseMarshalCachedObject** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*data, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Loosely marshals the passed DataStructure based object to the passed stream returning nothing.*

- virtual **commands::DataStructure** \* **looseUnmarshalCachedObject** (**OpenWireFormat** \*wireFormat, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Loose Unmarshal the cached object.*

- virtual int **tightMarshalNestedObject1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*object, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.*

- virtual void **tightMarshalNestedObject2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*object, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Tightly marshals the passed DataStructure based object to the passed streams returning nothing.*

- virtual **commands::DataStructure** \* **tightUnmarshalNestedObject** (**OpenWireFormat** \*wireFormat, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Tight Unmarshal the nested object.*

- virtual **commands::DataStructure** \* **looseUnmarshalNestedObject** (**OpenWireFormat** \*wireFormat, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Loose Unmarshal the nested object.*

- virtual void **looseMarshalNestedObject** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*object, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Loose marshall the nested object.*

- virtual std::string **tightUnmarshalString** (**decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Performs Tight Unmarshaling of String Objects.*

- virtual int **tightMarshalString1** (const std::string &value, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Tight Marshals the String to a Booleans Stream Object, returns the marshaled size.*

- virtual void **tightMarshalString2** (const std::string &value, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Tight Marshals the passed string to the streams passed.*
- virtual void **looseMarshalString** (const std::string value, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Loose Marshal the String to the DataOutputStream passed.*
- virtual std::string **looseUnmarshalString** (**decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Loose Un-Marshal the String to the DataOutputStream passed.*
- virtual int **tightMarshalLong1** (**OpenWireFormat** \*wireFormat, long long value, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Tightly marshal the long long to the BooleanStream passed.*
- virtual void **tightMarshalLong2** (**OpenWireFormat** \*wireFormat, long long value, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Tightly marshal the long long to the Streams passed.*
- virtual long long **tightUnmarshalLong** (**OpenWireFormat** \*wireFormat, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Tight marshal the long long type.*
- virtual void **looseMarshalLong** (**OpenWireFormat** \*wireFormat, long long value, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Tightly marshal the long long to the BooleanStream passed.*
- virtual long long **looseUnmarshalLong** (**OpenWireFormat** \*wireFormat, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Loose marshal the long long type.*
- virtual std::vector< unsigned char > **tightUnmarshalByteArray** (**decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Tight Unmarshal an array of char.*
- virtual std::vector< unsigned char > **looseUnmarshalByteArray** (**decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Loose Unmarshal an array of char.*
- virtual std::vector< unsigned char > **tightUnmarshalConstByteArray** (**decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs, int size) throw ( decaf::io::IOException )  
*Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.*

- virtual `std::vector< unsigned char > looseUnmarshalConstByteArray (decaf::io::DataInputStream *dataIn, int size) throw ( decaf::io::IOException )`  
*Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.*
- virtual `commands::DataStructure * tightUnmarshalBrokerError (OpenWireFormat *wireFormat, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Tight Unmarshal the Error object.*
- virtual `int tightMarshalBrokerError1 (OpenWireFormat *wireFormat, commands::DataStructure *data, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Tight Marshal the Error object.*
- virtual `void tightMarshalBrokerError2 (OpenWireFormat *wireFormat, commands::DataStructure *data, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Tight Marshal the Error object.*
- virtual `commands::DataStructure * looseUnmarshalBrokerError (OpenWireFormat *wireFormat, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Loose Unmarshal the Error object.*
- virtual `void looseMarshalBrokerError (OpenWireFormat *wireFormat, commands::DataStructure *data, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Tight Marshal the Error object.*
- `template<typename T >`  
`int tightMarshalObjectArray1 (OpenWireFormat *wireFormat, std::vector< T > objects, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Tightly Marshal an array of DataStructure objects to the provided boolean stream, and return the size that the tight marshalling is going to take.*
- `template<typename T >`  
`void tightMarshalObjectArray2 (OpenWireFormat *wireFormat, std::vector< T > objects, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Tightly Marshal an array of DataStructure objects to the provided boolean stream and data output stream.*
- `template<typename T >`  
`void looseMarshalObjectArray (OpenWireFormat *wireFormat, std::vector< T > objects, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`

*Loosely Marshal an array of DataStructure objects to the provided boolean stream and data output stream.*

- virtual std::string **readAsciiString** (decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )

*Given an DataInputStream read a know ASCII formatted string from the input and return that string.*

### 6.132.1 Detailed Description

Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol.

Since

2.0

### 6.132.2 Constructor & Destructor Documentation

- 6.132.2.1 virtual activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::~BaseDataStreamMarshaller ( ) [inline, virtual]

### 6.132.3 Member Function Documentation

- 6.132.3.1 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshal ( OpenWireFormat \*format *AMQCPP\_UNUSED*, commands::DataStructure \*command *AMQCPP\_UNUSED*, decaf::io::DataOutputStream \*ds *AMQCPP\_UNUSED* ) throw ( decaf::io::IOException ) [inline, virtual]

Tight Marshal to the given stream.

#### Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.132.3.2 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalBrokerError ( OpenWireFormat * wireFormat, commands::DataStructure * data, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )`  
[protected, virtual]

Tight Marshal the Error object.

#### Parameters

<i>wireFormat</i>	- The OpenwireFormat properties
<i>data</i>	- Error to Marshal
<i>dataOut</i>	- stream to write marshalled data to

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.132.3.3 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalCachedObject ( OpenWireFormat * wireFormat, commands::DataStructure * data, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )`  
[protected, virtual]

Loosely marshals the passed DataStructure based object to the passed stream returning nothing.

#### Parameters

<i>wireFormat</i>	- The OpenwireFormat properties
<i>data</i>	- DataStructure Object Pointer to marshal
<i>dataOut</i>	- stream to write marshaled data to

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.132.3.4 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalLong ( OpenWireFormat * wireFormat, long long value, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )`  
[protected, virtual]

Tightly marshal the long long to the BooleanStream passed.

#### Parameters

<i>wireFormat</i>	- The OpenwireFormat properties
<i>value</i>	- long long to marshal
<i>dataOut</i>	- DataOutputStream to marshal to.

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.5** virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalNestedObject ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *object*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException )  
[protected, virtual]

Loose marshall the nested object.

**Parameters**

<i>wireFormat</i>	- The OpenwireFormat properties
<i>object</i>	- DataStructure Object Pointer to marshal
<i>dataOut</i>	- stream to write marshaled data to

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.6** template<typename T> void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, std::vector< T > *objects*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException )  
[inline, protected]

Loosely Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

**Parameters**

<i>wireFormat</i>	- The OpenwireFormat properties
<i>objects</i>	- array of DataStructure object pointers.
<i>dataOut</i>	- stream to write marshalled data to

**Returns**

size of the marshalled data

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

References AMQ\_CATCH\_EXCEPTION\_CONVERT, AMQ\_CATCH\_RETHROW, and AMQ\_CATCHALL\_THROW.

**6.132.3.7** virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalString  
( const std::string *value*, decaf::io::DataOutputStream \* *dataOut* ) throw (  
decaf::io::IOException ) [protected, virtual]

Loose Marshal the String to the DataOutputStream passed.

#### Parameters

<i>value</i>	- string to marshal
<i>dataOut</i>	- stream to write marshaled form to

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.8** virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshal  
( OpenWireFormat \**format* *AMQCPP\_UNUSED*, commands::DataStructure  
\**command* *AMQCPP\_UNUSED*, decaf::io::DataInputStream \**dis*  
*AMQCPP\_UNUSED* ) throw ( decaf::io::IOException ) [inline,  
virtual]

Loose Un-Marshall to the given stream.

#### Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshall
<i>dis</i>	- the DataInputStream to Un-Marshall from

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.9** virtual commands::DataStructure\* ac-  
tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalBrokerError  
( OpenWireFormat \* *wireFormat*, decaf::io::DataInputStream \* *dataIn* )  
throw ( decaf::io::IOException ) [protected, virtual]

Loose Unmarshal the Error object.

#### Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>dataIn</i>	- stream to read marshalled form from

#### Returns

pointer to a new DataStructure Object

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.10** `virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalByteArray ( decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
[protected, virtual]

Loose Unmarshal an array of char.

**Parameters**

<i>dataIn</i>	- the DataInputStream to Un-Marshall from
---------------	---

**Returns**

the unmarshalled vector of chars.

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.11** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalCachedObject ( OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
[protected, virtual]

Loose Unmarshal the cached object.

**Parameters**

<i>wireFormat</i>	- The OpenwireFormat properties
<i>dataIn</i>	- stream to read marshaled form from

**Returns**

pointer to a new DataStructure Object

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------



**6.132.3.12** `virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalConstByteArray ( decaf::io::DataInputStream * dataIn, int size ) throw ( decaf::io::IOException )` [protected, virtual]

Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

#### Parameters

<i>dataIn</i>	- the DataInputStream to Un-Marshal from
<i>size</i>	- size of the const array to unmarshal

#### Returns

the unmarshaled vector of chars.

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.13** `virtual long long activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalLong ( OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [protected, virtual]

Loose marshal the long long type.

#### Parameters

<i>wireFormat</i>	- The OpenwireFormat properties
<i>dataIn</i>	- stream to read marshaled form from

#### Returns

the unmarshaled long long

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.14** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalNestedObject ( OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [protected, virtual]

Loose Unmarshal the nested object.

#### Parameters

<i>wireFormat</i>	- The OpenwireFormat properties
-------------------	---------------------------------

<i>dataIn</i>	- stream to read marshaled form from
---------------	--------------------------------------

**Returns**

pointer to a new DataStructure Object

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.15** `virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalString ( decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
[protected, virtual]

Loose Un-Marshal the String to the DataOuputStream passed.

**Parameters**

<i>dataIn</i>	- stream to read marshaled form from
---------------	--------------------------------------

**Returns**

the unmarshaled string

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.16** `virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::readAsciiString ( decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
[protected, virtual]

Given an DataInputStream read a know ASCII formatted string from the input and return that string.

**Parameters**

<i>dataIn</i>	- DataInputStream to read from
---------------	--------------------------------

**Returns**

string value read from stream

6.132.3.17 `virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshal1 ( OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED ) throw ( decaf::io::IOException ) [inline, virtual]`

Tight Marshal to the given stream.

#### Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>bs</i>	- boolean stream to marshal to.

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.132.3.18 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshal2 ( OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataOutputStream *ds AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED ) throw ( decaf::io::IOException ) [inline, virtual]`

Tight Marshal to the given stream.

#### Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.132.3.19 `virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalBrokerError1 ( OpenWireFormat * wireFormat, commands::DataStructure * data, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [protected, virtual]`

Tight Marshal the Error object.

#### Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>data</i>	- Error to Marshal
<i>bs</i>	- boolean stream to marshal to.

**Returns**

size of the marshalled data

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.20** `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalBrokerError2 ( OpenWireFormat * wireFormat, commands::DataStructure * data, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [protected, virtual]

Tight Marshal the Error object.

**Parameters**

<i>wireFormat</i>	- The OpenwireFormat properties
<i>data</i>	- Error to Marshal
<i>dataOut</i>	- stream to write marshalled data to
<i>bs</i>	- boolean stream to marshal to.

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.21** `virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalCachedObject1 ( OpenWireFormat * wireFormat, commands::DataStructure * data, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.

**Parameters**

<i>wireFormat</i>	- The OpenwireFormat properties
<i>data</i>	- DataStructure Object Pointer to marshal
<i>bs</i>	- boolean stream to marshal to.

**Returns**

size of data written.

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.22** virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalCachedObject2  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *data*,  
decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* )  
throw ( decaf::io::IOException ) [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed streams returning nothing.

#### Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>data</i>	- DataStructure Object Pointer to marshal
<i>bs</i>	- boolean stream to marshal to.
<i>dataOut</i>	- stream to write marshaled data to

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.23** virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalLong1  
( OpenWireFormat \* *wireFormat*, long long *value*, utils::BooleanStream \*  
*bs* ) throw ( decaf::io::IOException ) [protected, virtual]

Tightly marshal the long long to the BooleanStream passed.

#### Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>value</i>	- long long to marshal
<i>bs</i>	- boolean stream to marshal to.

#### Returns

size of data written.

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.24** virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalLong2  
( OpenWireFormat \* *wireFormat*, long long *value*,  
decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* )  
throw ( decaf::io::IOException ) [protected, virtual]

Tightly marshal the long long to the Streams passed.

#### Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
-------------------	---------------------------------

<i>value</i>	- long long to marshal
<i>dataOut</i>	- stream to write marshaled form to
<i>bs</i>	- boolean stream to marshal to.

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.25** `virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalNestedObject1 ( OpenWireFormat * wireFormat, commands::DataStructure * object, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[protected, virtual]

Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.

**Parameters**

<i>wireFormat</i>	- The OpenwireFormat properties
<i>object</i>	- DataStructure Object Pointer to marshal
<i>bs</i>	- boolean stream to marshal to.

**Returns**

size of data written.

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.26** `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalNestedObject2 ( OpenWireFormat * wireFormat, commands::DataStructure * object, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[protected, virtual]

Tightly marshals the passed DataStructure based object to the passed streams returning nothing.

**Parameters**

<i>wireFormat</i>	- The OpenwireFormat properties
<i>object</i>	- DataStructure Object Pointer to marshal
<i>bs</i>	- boolean stream to marshal to.
<i>dataOut</i>	- stream to write marshaled data to

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.27** `template<typename T> int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1 ( OpenWireFormat * wireFormat, std::vector< T > objects, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [inline, protected]`

Tightly Marshal an array of DataStructure objects to the provided boolean stream, and return the size that the tight marshalling is going to take.

#### Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>objects</i>	- array of DataStructure object pointers.
<i>bs</i>	- boolean stream to marshal to.

#### Returns

size of the marshalled data

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

References AMQ\_CATCH\_EXCEPTION\_CONVERT, AMQ\_CATCH\_RETHROW, and AMQ\_CATCHALL\_THROW.

**6.132.3.28** `template<typename T> void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2 ( OpenWireFormat * wireFormat, std::vector< T > objects, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [inline, protected]`

Tightly Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

#### Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>objects</i>	- array of DataStructure object pointers.
<i>dataOut</i>	- stream to write marshalled data to
<i>bs</i>	- boolean stream to marshal to.

#### Returns

size of the marshalled data

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

References AMQ\_CATCH\_EXCEPTION\_CONVERT, AMQ\_CATCH\_RETHROW, and AMQ\_CATCHALL\_THROW.

**6.132.3.29** `virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalString1 ( const std::string & value, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [protected, virtual]`

Tight Marshals the String to a Booleans Stream Object, returns the marshaled size.

#### Parameters

<i>value</i>	- string to marshal
<i>bs</i>	- BooleanStream to use.

#### Returns

size of marshaled string.

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.30** `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalString2 ( const std::string & value, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [protected, virtual]`

Tight Marshals the passed string to the streams passed.

#### Parameters

<i>value</i>	- string to marshal
<i>dataOut</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.31** `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshal ( OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataInputStream *dis AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED ) throw ( decaf::io::IOException ) [inline, virtual]`

Tight Un-Marshal to the given stream.

#### Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to Un-Marshal from.



## Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.32** virtual **commands::DataStructure\*** **activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalBrokerError** ( **OpenWireFormat** \* *wireFormat*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** )  
[protected, virtual]

Tight Unarshall the Error object.

## Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>dataIn</i>	- stream to read marshalled form from
<i>bs</i>	- boolean stream to marshal to.

## Returns

pointer to a new DataStructure Object

## Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.33** virtual **std::vector<unsigned char>** **activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalByteArray** ( **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [protected, virtual]

Tight Unmarshal an array of char.

## Parameters

<i>dataIn</i>	- the DataInputStream to Un-Marshall from
<i>bs</i>	- boolean stream to unmarshal from.

## Returns

the unmarshaled vector of chars.

## Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.34** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalCachedObject ( OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [protected, virtual]

Tight Unmarshal the cached object.

#### Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>dataIn</i>	- stream to read marshaled form from
<i>bs</i>	- boolean stream to marshal to.

#### Returns

pointer to a new DataStructure Object

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.35** `virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalConstByteArray ( decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs, int size ) throw ( decaf::io::IOException )` [protected, virtual]

Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

#### Parameters

<i>dataIn</i>	- the DataInputStream to Un-Marshall from
<i>bs</i>	- boolean stream to unmarshal from.
<i>size</i>	- size of the const array to unmarshal

#### Returns

the unmarshaled vector of chars.

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.36** virtual long long activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalLong  
( OpenWireFormat \* *wireFormat*, decaf::io::DataInputStream \*  
*dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[protected, virtual]

Tight marshal the long long type.

#### Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>dataIn</i>	- stream to read marshaled form from
<i>bs</i>	- boolean stream to marshal to.

#### Returns

the unmarshaled long long

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.37** virtual commands::DataStructure\* ac-  
tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalNestedObject  
( OpenWireFormat \* *wireFormat*, decaf::io::DataInputStream \*  
*dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[protected, virtual]

Tight Unmarshal the nested object.

#### Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>dataIn</i>	- stream to read marshaled form from
<i>bs</i>	- boolean stream to marshal to.

#### Returns

pointer to a new DataStructure Object

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.38** virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalString  
( decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw  
( decaf::io::IOException ) [protected, virtual]

Performs Tight Unmarshaling of String Objects.

**Parameters**

<i>dataIn</i>	- the DataInputStream to Un-Marshall from
<i>bs</i>	- boolean stream to unmarshal from.

**Returns**

the unmarshaled string.

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.132.3.39** `static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toHexFromBytes  
( const std::vector< unsigned char > & data ) [static]`

given an array of bytes, convert that array to a Hexidecimal coded string that represents that data.

**Parameters**

<i>data</i>	- unsigned char data array pointer
-------------	------------------------------------

**Returns**

a string coded in hex that represents the data

**6.132.3.40** `static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString  
( const commands::TransactionId * txnId ) [static]`

Converts the given transaction ID into a String.

**Parameters**

<i>txnId</i>	- TransactionId poitner
--------------	-------------------------

**Returns**

string representation of the id

**6.132.3.41** `static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString  
( const commands::ProducerId * id ) [static]`

Converts the object to a String.

**Parameters**

<i>id</i>	- ProducerId pointer
-----------	----------------------

**Returns**

string representing the id

**6.132.3.42** static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString  
( const commands::MessageId \* *id* ) [static]

Converts the object to a String.

**Parameters**

<i>id</i>	- MessageId pointer
-----------	---------------------

**Returns**

string representing the id

Referenced by activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getCMSMessageID().

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/**BaseDataStreamMarshaller.h**

**6.133 activemq::commands::BaseDataStructure Class Reference**

```
#include <src/main/activemq/commands/BaseDataStructure.h>
```

Inheritance diagram for activemq::commands::BaseDataStructure:

**Public Member Functions**

- virtual ~**BaseDataStructure** ()
- virtual bool **isMarshalAware** () const  
*Determine if this object is aware of marshaling and should have its before and after marshaling methods called.*
- virtual void **beforeMarshal** (wireformat::WireFormat \*wireFormat AMQCPP\_UNUSED) throw ( decaf::io::IOException )  
*Perform any processing needed before an marshal.*
- virtual void **afterMarshal** (wireformat::WireFormat \*wireFormat AMQCPP\_UNUSED) throw ( decaf::io::IOException )  
*Perform any processing needed after an unmarshal.*

- virtual void **beforeUnmarshal** (**wireformat::WireFormat** \*wireFormat AMQCPP\_UNUSED) throw ( decaf::io::IOException )  
*Perform any processing needed before an unmarshal.*
- virtual void **afterUnmarshal** (**wireformat::WireFormat** \*wireFormat AMQCPP\_UNUSED) throw ( decaf::io::IOException )  
*Perform any processing needed after an unmarshal.*
- virtual void **setMarshaledForm** (**wireformat::WireFormat** \*wireFormat AMQCPP\_UNUSED, const std::vector< char > &data AMQCPP\_UNUSED)  
*Called to set the data to this object that will contain the objects marshaled form.*
- virtual std::vector< unsigned char > **getMarshaledForm** (**wireformat::WireFormat** \*wireFormat AMQCPP\_UNUSED)  
*Called to get the data to this object that will contain the objects marshaled form.*
- virtual void **copyDataStructure** (const **DataStructure** \*src AMQCPP\_UNUSED)  
  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value AMQCPP\_UNUSED) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*

### 6.133.1 Constructor & Destructor Documentation

- 6.133.1.1 virtual activemq::commands::BaseDataStructure::~BaseDataStructure ( )  
[inline, virtual]

### 6.133.2 Member Function Documentation

- 6.133.2.1 virtual void activemq::commands::BaseDataStructure::afterMarshal ( **wireformat::WireFormat** \*wireFormat **AMQCPP\_UNUSED** ) throw ( **decaf::io::IOException** ) [inline, virtual]

Perform any processing needed after an unmarshal.

#### Parameters

<i>wireformat</i>	- the OpenWireFormat object in use.
-------------------	-------------------------------------

**6.133.2.2** `virtual void activemq::commands::BaseDataStructure::afterUnmarshal ( wireformat::WireFormat *wireFormat AMQCPP_UNUSED ) throw ( decaf::io::IOException ) [inline, virtual]`

Perform any processing needed after an unmarshal.

#### Parameters

<i>wireformat</i>	- the OpenWireFormat object in use.
-------------------	-------------------------------------

Reimplemented in **activemq::commands::Message** (p. 2601), and **activemq::commands::WireFormatInfo** (p. 4096).

**6.133.2.3** `virtual void activemq::commands::BaseDataStructure::beforeMarshal ( wireformat::WireFormat *wireFormat AMQCPP_UNUSED ) throw ( decaf::io::IOException ) [inline, virtual]`

Perform any processing needed before an marshal.

#### Parameters

<i>wireformat</i>	- the OpenWireFormat object in use.
-------------------	-------------------------------------

Reimplemented in **activemq::commands::Message** (p. 2601), and **activemq::commands::WireFormatInfo** (p. 4097).

**6.133.2.4** `virtual void activemq::commands::BaseDataStructure::beforeUnmarshal ( wireformat::WireFormat *wireFormat AMQCPP_UNUSED ) throw ( decaf::io::IOException ) [inline, virtual]`

Perform any processing needed before an unmarshal.

#### Parameters

<i>wireformat</i>	- the OpenWireFormat object in use.
-------------------	-------------------------------------

**6.133.2.5** `virtual void activemq::commands::BaseDataStructure::copyDataStructure ( const DataStructure *src AMQCPP_UNUSED ) [inline, virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented in **activemq::commands::BooleanExpression** (p. 862).

**6.133.2.6** `virtual bool activemq::commands::BaseDataStructure::equals ( const  
DataStructure *value AMQCPP_UNUSED ) const` `[inline, virtual]`

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Referenced by `activemq::commands::BooleanExpression::equals()`, and `activemq::commands::BaseCommand::equals()`.

**6.133.2.7** `virtual std::vector<unsigned char> ac-  
tivemq::commands::BaseDataStructure::getMarshaledForm (   
wireformat::WireFormat *wireFormat AMQCPP_UNUSED )` `[inline, virtual]`

Called to get the data to this object that will contain the objects marshaled form.

#### Parameters

<i>wireFormat</i>	- the wireformat object to control unmarshaling
-------------------	---

#### Returns

buffer that holds the objects data.

**6.133.2.8** `virtual bool activemq::commands::BaseDataStructure::isMarshalAware ( ) const` `[inline, virtual]`

Determine if this object is aware of marshaling and should have its before and after marshaling methods called.

Defaults to false.

#### Returns

true if aware of marshaling

Implements `activemq::wireformat::MarshalAware` (p. 2566).

Reimplemented in `activemq::commands::ActiveMQMapMessage` (p. 359), `activemq::commands::Message` (p. 2607), and `activemq::commands::WireFormatInfo` (p. 4100).



**6.133.2.9** `virtual void activemq::commands::BaseDataStructure::setMarshaledForm ( wireformat::WireFormat *wireFormat AMQCPP_UNUSED, const std::vector< char > &data AMQCPP_UNUSED ) [inline, virtual]`

Called to set the data to this object that will contain the objects marshaled form.

#### Parameters

<i>wireFormat</i>	- the wireformat object to control unmarshaling
<i>data</i>	- vector of object binary data

**6.133.2.10** `virtual std::string activemq::commands::BaseDataStructure::toString ( ) const [inline, virtual]`

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Implements **activemq::commands::DataStructure** (p. 1718).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 189), **activemq::commands::ActiveMQBytesMessage** (p. 229), **activemq::commands::ActiveMQDestination** (p. 321), **activemq::commands::ActiveMQMapMessage** (p. 364), **activemq::commands::ActiveMQMessage** (p. 392), **activemq::commands::ActiveMQObjectMessage** (p. 440), **activemq::commands::ActiveMQQueue** (p. 483), **activemq::commands::ActiveMQStreamMessage** (p. 547), **activemq::commands::ActiveMQTempDestination** (p. 582), **activemq::commands::ActiveMQTempQueue** (p. 611), **activemq::commands::ActiveMQTempTopic** (p. 641), **activemq::commands::ActiveMQTextMessage** (p. 671), **activemq::commands::ActiveMQTopic** (p. 701), **activemq::commands::BaseCommand** (p. 770), **activemq::commands::BooleanExpression** (p. 863), **activemq::commands::BrokerId** (p. 877), **activemq::commands::BrokerInfo** (p. 908), **activemq::commands::Command** (p. 1231), **activemq::commands::ConnectionControl** (p. 1305), **activemq::commands::ConnectionError** (p. 1335), **activemq::commands::ConnectionId** (p. 1368), **activemq::commands::ConnectionInfo** (p. 1398), **activemq::commands::ConsumerControl** (p. 1445), **activemq::commands::ConsumerId** (p. 1476), **activemq::commands::ConsumerInfo** (p. 1507), **activemq::commands::ControlCommand** (p. 1538), **activemq::commands::DataArrayResponse** (p. 1574), **activemq::commands::DataResponse** (p. 1634), **activemq::commands::DestinationInfo** (p. 1783), **activemq::commands::DiscoveryEvent** (p. 1814), **activemq::commands::ExceptionResponse** (p. 1897), **activemq::commands::FlushCommand** (p. 2001), **activemq::commands::IntegerResponse** (p. 2162), **activemq::commands::JournalQueueAck** (p. 2227), **activemq::commands::JournalTopicAck** (p. 2255), **activemq::commands::JournalTrace** (p. 2283), **activemq::commands::JournalTransaction** (p. 2310), **activemq::commands::KeepAliveInfo** (p. 2337), **activemq::commands::LastPartialCommand** (p. 2373), **activemq::commands::LocalTransactionId** (p. 2424), **activemq::commands::Message** (p. 2611), **activemq::commands::MessageAck** (p. 2647), **activemq::commands::MessageDispatch** (p. 2682), **activemq::commands::MessageDispatchNotification** (p. 2719), **activemq::commands::MessageId** (p. 2754), **activemq::commands::MessagePull** (p. 2827), **activemq::commands::NetworkBridgeFilter** (p. 2880), **activemq::commands::PartialCommand** (p. 3005), **activemq::commands::ProducerAck** (p. 3127), **activemq::commands::ProducerId** (p. 3160), **activemq::commands::ProducerInfo** (p. 3189), **activemq::commands::RemoveInfo**

(p. 3287), **activemq::commands::RemoveSubscriptionInfo** (p. 3317), **activemq::commands::ReplayComm** (p. 3346), **activemq::commands::Response** (p. 3382), **activemq::commands::SessionId** (p. 3480), **activemq::commands::SessionInfo** (p. 3507), **activemq::commands::ShutdownInfo** (p. 3575), **activemq::commands::SubscriptionInfo** (p. 3785), **activemq::commands::TransactionId** (p. 3935), **activemq::commands::TransactionInfo** (p. 3963), **activemq::commands::WireFormatInfo** (p. 4104), and **activemq::commands::XATransactionId** (p. 4146).

Referenced by **activemq::commands::BooleanExpression::toString()**, and **activemq::commands::BaseCommand**.

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BaseDataStructure.h`

## 6.134 **binary\_function** Class Reference

Inheritance diagram for **binary\_function**:

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Comparator.h`

## 6.135 **decaf::net::BindException** Class Reference

```
#include <src/main/decaf/net/BindException.h>
```

Inheritance diagram for **decaf::net::BindException**:

### Public Member Functions

- **BindException** () throw ()  
*Default Constructor.*
- **BindException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **BindException** (const **BindException** &ex) throw ()  
*Copy Constructor.*
- **BindException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*

- **BindException** (const std::exception \***cause**) throw ()  
*Constructor.*
- **BindException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **BindException** \* **clone** () const  
*Clones this exception.*
- virtual ~**BindException** () throw ()

### 6.135.1 Constructor & Destructor Documentation

6.135.1.1 decaf::net::BindException::BindException ( ) throw () [inline]

Default Constructor.

6.135.1.2 decaf::net::BindException::BindException ( const Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.135.1.3 decaf::net::BindException::BindException ( const BindException & ex ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.135.1.4 decaf::net::BindException::BindException ( const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
-------------	--------------------------------------

<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.135.1.5** `decaf::net::BindException::BindException ( const std::exception * cause ) throw ()`  
`[inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.135.1.6** `decaf::net::BindException::BindException ( const char * file, const int lineNumber,  
const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.135.1.7** `virtual decaf::net::BindException::~~BindException ( ) throw ()` `[inline, virtual]`

## 6.135.2 Member Function Documentation

**6.135.2.1** `virtual BindException* decaf::net::BindException::clone ( ) const` `[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 3629).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**BindException.h**

## 6.136 decaf::io::BlockingByteArrayInputStream Class Reference

This is a blocking version of a byte buffer stream.

```
#include <src/main/decaf/io/BlockingByteArrayInputStream.h>
```

Inheritance diagram for decaf::io::BlockingByteArrayInputStream:

### Public Member Functions

- **BlockingByteArrayInputStream** ()  
*Default Constructor - uses a default internal buffer.*
- **BlockingByteArrayInputStream** (const unsigned char \*buffer, int bufferSize)

*Constructor that initializes the internal buffer.*

- virtual ~**BlockingByteArrayInputStream** ()
- virtual void **setByteArray** (const unsigned char \*buffer, int bufferSize)

- virtual int **available** () const throw ( decaf::io::IOException )

*Indicates the number of bytes available.*

*The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute. The default implementation of this method returns zero.*

#### Returns

*the number of bytes available on this input stream.*

#### Exceptions

<b>IOException</b> (p. 2210)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

- virtual void **close** () throw ( decaf::io::IOException )  
*Closes the **InputStream** (p. 2105) freeing any resources that might have been acquired during the lifetime of this stream.*  
*The default implementation of this method does nothing.*

- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Skips over and discards  $n$  bytes of data from this input stream.*

*The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before  $n$  bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.*

*The skip method of **InputStream** (p. 2105) creates a byte array and then repeatedly reads into it until  $num$  bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*

#### Parameters

num	The number of bytes to skip.
-----	------------------------------

#### Returns

*total bytes skipped*

#### Exceptions

<b>IOException</b> (p. 2210)	<i>if an I/O error occurs.</i>
<b>UnsupportedOperationException</b>	<i>if the concrete stream class does not support skipping bytes.</i>

## Protected Member Functions

- virtual int **doReadByte** () throw ( **IOException** )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( **decaf::io::IOException**, **decaf::lang::exceptions::IndexOutOfBoundsException**, **decaf::lang::exceptions::NullPointerException** )

### 6.136.1 Detailed Description

This is a blocking version of a byte buffer stream. Read operations block until the requested data becomes available in the internal buffer via a call to **setByteArray**.

### 6.136.2 Constructor & Destructor Documentation

#### 6.136.2.1 **decaf::io::BlockingByteArrayInputStream::BlockingByteArrayInputStream** ( )

Default Constructor - uses a default internal buffer.

#### 6.136.2.2 **decaf::io::BlockingByteArrayInputStream::BlockingByteArrayInputStream** ( const unsigned char \* *buffer*, int *bufferSize* )

Constructor that initializes the internal buffer.

See also

**setByteArray** (p. 848).

6.136.2.3 `virtual decaf::io::BlockingByteArrayInputStream::~~BlockingByteArrayInputStream ( ) [virtual]`

### 6.136.3 Member Function Documentation

6.136.3.1 `virtual int decaf::io::BlockingByteArrayInputStream::available ( ) const throw ( decaf::io::IOException ) [virtual]`

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

#### Returns

the number of bytes available on this input stream.

#### Exceptions

<i>IOException</i> (p. 2210)	if an I/O error occurs.
---------------------------------	-------------------------

Reimplemented from **decaf::io::InputStream** (p. 2107).

6.136.3.2 `virtual void decaf::io::BlockingByteArrayInputStream::close ( ) throw ( decaf::io::IOException ) [virtual]`

Closes the **InputStream** (p. 2105) freeing any resources that might have been aquired during the lifetime of this stream.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::InputStream** (p. 2108).

6.136.3.3 `virtual int decaf::io::BlockingByteArrayInputStream::doReadArrayBounded ( unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException ) [protected, virtual]`

Reimplemented from **decaf::io::InputStream** (p. 2108).

6.136.3.4 `virtual int decaf::io::BlockingByteArrayInputStream::doReadByte ( ) throw ( IOException ) [protected, virtual]`

Implements **decaf::io::InputStream** (p. 2109).

6.136.3.5 `virtual void decaf::io::BlockingByteArrayInputStream::setByteArray ( const unsigned char * buffer, int bufferSize )` [virtual]

6.136.3.6 `virtual long long decaf::io::BlockingByteArrayInputStream::skip ( long long num ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )` [virtual]

Skips over and discards *n* bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2105) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

#### Returns

total bytes skipped

#### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
<i>UnsupportedOperationException</i>	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 2113).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/BlockingByteArrayInputStream.h`

## 6.137 decaf::util::concurrent::BlockingQueue< E > Class Template Reference

A **decaf::util::Queue** (p. 3239) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.

```
#include <src/main/decaf/util/concurrent/BlockingQueue.h>
```



## 6.137 decaf::util::concurrent::BlockingQueue< E > Class Template Reference 849

Inheritance diagram for decaf::util::concurrent::BlockingQueue< E >:

### Public Member Functions

- virtual **~BlockingQueue** ()
- virtual void **put** (const E &value)=0 throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )

*Inserts the specified element into this queue, waiting if necessary for space to become available.*

- virtual bool **offer** (const E &e, long timeout, const **TimeUnit** &unit)=0 throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )

*Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.*

- virtual E **take** ()=0 throw ( decaf::lang::exceptions::InterruptedException )

*Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.*

- virtual bool **poll** (E &result, long long timeout, const **TimeUnit** &unit)=0 throw ( decaf::lang::exceptions::InterruptedException )

*Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available.*

- virtual int **remainingCapacity** () const =0

*Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or Integer::MAX\_VALUE if there is no intrinsic limit.*

- virtual std::size\_t **drainTo** (**Collection**< E > &c)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException )

*Removes all available elements from this queue and adds them to the given collection.*

- virtual std::size\_t **drainTo** (**Collection**< E > &c, std::size\_t maxElements)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException )

*Removes at most the given number of available elements from this queue and adds them to the given collection.*

### 6.137.1 Detailed Description

`template<typename E> class decaf::util::concurrent::BlockingQueue< E >`

A **decaf::util::Queue** (p. 3239) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element. **BlockingQueue** (p. 848) methods come in four forms, with different ways of handling operations that cannot be satisfied immediately, but may be satisfied at some point in the future: one throws an exception, the second returns a special value (either `true` or `false`, depending on the operation), the third blocks the current thread indefinitely until the operation can succeed, and the fourth blocks for only a given maximum time limit before giving up. These methods are summarized in the following table:

	<i>Throws exception</i>	<i>Boolean Flag</i>	<i>Blocks</i>	<i>Times out</i>
<b>Insert</b>	<b>add(e)</b> (p. 177)	<b>offer(e)</b> (p. 853)	<b>put(e)</b> (p. 854)	<b>offer(e, time, unit)</b> (p. ??)
<b>Remove</b>	<b>remove()</b> (p. 179)	<b>poll()</b> (p. 854)	<b>take()</b> (p. 855)	<b>poll(time, unit)</b> (p. ??)
<b>Examine</b>	<b>element()</b> (p. 178)	<b>peek()</b> (p. 3242)	<i>not applicable</i>	<i>not applicable</i>

A **BlockingQueue** (p. 848) may be capacity bounded. At any given time it may have a `remainingCapacity` beyond which no additional elements can be put without blocking. A **BlockingQueue** (p. 848) without any intrinsic capacity constraints always reports a remaining capacity of `Integer::MAX_VALUE`.

**BlockingQueue** (p. 848) implementations are designed to be used primarily for producer-consumer queues, but additionally support **decaf::util::Collection** (p. 1216) interface. So, for example, it is possible to remove an arbitrary element from a queue using `remove(x)`. However, such operations are in general *not* performed very efficiently, and are intended for only occasional use, such as when a queued message is cancelled.

**BlockingQueue** (p. 848) implementations are thread-safe. All queuing methods achieve their effects atomically using internal locks or other forms of concurrency control. However, the *bulk* **Collection** (p. 1216) operations `addAll`, `containsAll`, `retainAll` and `removeAll` are *not* necessarily performed atomically unless specified otherwise in an implementation. So it is possible, for example, for `addAll(c)` to fail (throwing an exception) after adding only some of the elements in `c`.

A **BlockingQueue** (p. 848) does *not* intrinsically support any kind of "close" or "shutdown" operation to indicate that no more items will be added. The needs and usage of such features tend to be implementation-dependent. For example, a common tactic is for producers to insert special *end-of-stream* or *poison* objects, that are interpreted accordingly when taken by consumers.

Usage example, based on a typical producer-consumer scenario. Note that a **BlockingQueue** (p. 848) can safely be used with multiple producers and multiple consumers.

```
class Producer : public Runnable {
private:
```

### 6.137 decaf::util::concurrent::BlockingQueue< E > Class Template Reference 851

```
BlockingQueue* queue;

public:

    Producer( BlockingQueue* q ) : queue( q ) {}

    virtual void run() {
        try {
            while( true ) { queue->put( produce() ); }
        } catch( InterruptedException& ex ) { ... handle ...}
    }

    Object produce() { ... }
}

class Consumer : public Runnable {
private:

    BlockingQueue* queue;

public:

    Consumer( BlockingQueue* q ) : queue( q ) {}

    virtual void run() {
        try {
            while( true ) { consume( queue->take() (p.855) ); }
        } catch( InterruptedException& ex ) { ... handle ...}
    }

    void consume( Object& x ) { ... }
}

int main( int argc, char** argv ) {

    BlockingQueue (p.848) q = new SomeQueueImplementation();
    Producer p( &q );
    Consumer c1( &q );
    Consumer c2( &q );
    Thread t1( &p ).start();
    Thread t2( &c1 ).start();
    Thread t3( &c2 ).start();
}
```

Memory consistency effects: As with other concurrent collections, actions in a thread prior to placing an object into a **BlockingQueue** (p.848) *happen-before* actions subsequent to the access or removal of that element from the **BlockingQueue** (p.848) in another thread.

Since

1.0

### 6.137.2 Constructor & Destructor Documentation

6.137.2.1 `template<typename E> virtual decaf::util::concurrent::BlockingQueue< E >::~~BlockingQueue( ) [inline, virtual]`

### 6.137.3 Member Function Documentation

6.137.3.1 `template<typename E> virtual std::size_t decaf::util::concurrent::BlockingQueue< E >::drainTo( Collection< E > & c ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException ) [pure virtual]`

Removes all available elements from this queue and adds them to the given collection.

This operation may be more efficient than repeatedly polling this queue. A failure encountered while attempting to add elements to collection *c* may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

#### Parameters

<i>c</i>	the collection to transfer elements into
----------	--

#### Returns

the number of elements transferred

#### Exceptions

<i>UnsupportedOperationException</i>	if addition of elements is not supported by the specified collection
<i>IllegalArgumentException</i>	if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3831).

6.137.3.2 `template<typename E> virtual std::size_t decaf::util::concurrent::BlockingQueue< E >::drainTo( Collection< E > & c, std::size_t maxElements ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException ) [pure virtual]`

Removes at most the given number of available elements from this queue and adds them to the given collection.

A failure encountered while attempting to add elements to collection *c* may result in

### 6.137 `decaf::util::concurrent::BlockingQueue< E >` Class Template Reference 853

elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

#### Parameters

<i>c</i>	the collection to transfer elements into
<i>maxElements</i>	the maximum number of elements to transfer

#### Returns

the number of elements transferred

#### Exceptions

<i>UnsupportedOperationException</i>	if addition of elements is not supported by the specified collection
<i>IllegalArgumentException</i>	if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3832).

```
6.137.3.3 template<typename E> virtual bool
decaf::util::concurrent::BlockingQueue< E >::offer
( const E & e, long timeout, const TimeUnit & unit )
throw ( decaf::lang::exceptions::InterruptedException,
decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException ) [pure
virtual]
```

Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.

#### Parameters

<i>e</i>	the element to add
<i>timeout</i>	how long to wait before giving up, in units of <i>unit</i>
<i>unit</i>	a <b>TimeUnit</b> (p. 3919) determining how to interpret the <i>timeout</i> parameter

#### Returns

`true` if successful, or `false` if the specified waiting time elapses before space is available

#### Exceptions

<i>InterruptedException</i>	if interrupted while waiting
-----------------------------	------------------------------

<i>NullPointerException</i>	if the specified element is null
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this queue

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3833).

```
6.137.3.4  template<typename E > virtual bool
           decaf::util::concurrent::BlockingQueue< E >::poll ( E
           & result, long long timeout, const TimeUnit & unit ) throw (
           decaf::lang::exceptions::InterruptedException ) [pure virtual]
```

Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available.

#### Parameters

<i>result</i>	the referenced value that will be assigned the value retrieved from the <b>Queue</b> (p. 3239). Undefined if this methods returned false.
<i>timeout</i>	how long to wait before giving up, in units of <i>unit</i>
<i>unit</i>	a <b>TimeUnit</b> (p. 3919) determining how to interpret the <i>timeout</i> parameter.

#### Returns

`true` if successful or `false` if the specified waiting time elapses before an element is available.

#### Exceptions

<i>InterruptedException</i>	if interrupted while waiting
-----------------------------	------------------------------

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3834).

```
6.137.3.5  template<typename E > virtual void
           decaf::util::concurrent::BlockingQueue< E >::put (
           const E & value ) throw ( decaf::lang::exceptions::InterruptedException,
           decaf::lang::exceptions::NullPointerException,
           decaf::lang::exceptions::IllegalArgumentException ) [pure
           virtual]
```

Inserts the specified element into this queue, waiting if necessary for space to become available.

#### Parameters

<i>value</i>	the element to add
--------------	--------------------

## 6.137 `decaf::util::concurrent::BlockingQueue< E >` Class Template Reference 855

### Exceptions

<i>InterruptedException</i>	if interrupted while waiting
<i>NullPointerException</i>	if the specified element is null
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this queue

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3835).

**6.137.3.6** `template<typename E> virtual int decaf::util::concurrent::BlockingQueue< E >::remainingCapacity ( ) const` [pure virtual]

Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or `Integer::MAX_VALUE` if there is no intrinsic limit.

Note that you *cannot* always tell if an attempt to insert an element will succeed by inspecting `remainingCapacity` because it may be the case that another thread is about to insert or remove an element.

### Returns

the remaining capacity

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3836).

**6.137.3.7** `template<typename E> virtual E decaf::util::concurrent::BlockingQueue< E >::take ( ) throw ( decaf::lang::exceptions::InterruptedException )` [pure virtual]

Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.

### Returns

the head of this queue

### Exceptions

<i>InterruptedException</i>	if interrupted while waiting
-----------------------------	------------------------------

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3837).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/BlockingQueue.h`

## 6.138 decaf::lang::Boolean Class Reference

```
#include <src/main/decaf/lang/Boolean.h>
```

Inheritance diagram for decaf::lang::Boolean:

### Public Member Functions

- **Boolean** (bool value)
- **Boolean** (const std::string &value)
- virtual  $\sim$ **Boolean** ()
- bool **booleanValue** () const
- std::string **toString** () const
- virtual int **compareTo** (const **Boolean** &b) const  
*Compares this **Boolean** (p. 856) instance with another.*
- virtual bool **operator==** (const **Boolean** &value) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Boolean** &value) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- bool **equals** (const **Boolean** &b) const
- virtual int **compareTo** (const bool &b) const  
*Compares this **Boolean** (p. 856) instance with another.*
- virtual bool **operator==** (const bool &value) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const bool &value) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- bool **equals** (const bool &b) const

### Static Public Member Functions

- static **Boolean valueOf** (bool value)
- static **Boolean valueOf** (const std::string &value)
- static bool **parseBoolean** (const std::string &value)  
*Parses the **String** (p. 3775) passed and extracts an bool.*
- static std::string **toString** (bool value)  
*Converts the bool to a **String** (p. 3775) representation.*



## Static Public Attributes

- static const **Boolean** **\_FALSE**  
*The Class object representing the primitive false boolean.*
- static const **Boolean** **\_TRUE**  
*The Class object representing the primitive type boolean.*

## 6.138.1 Constructor & Destructor Documentation

### 6.138.1.1 decaf::lang::Boolean::Boolean ( bool value )

#### Parameters

<i>value</i>	- primitive boolean to wrap.
--------------	------------------------------

### 6.138.1.2 decaf::lang::Boolean::Boolean ( const std::string & value )

#### Parameters

<i>value</i>	- <b>String</b> (p. 3775) value to convert to a boolean.
--------------	--

### 6.138.1.3 virtual decaf::lang::Boolean::~~Boolean ( ) [inline, virtual]

## 6.138.2 Member Function Documentation

### 6.138.2.1 bool decaf::lang::Boolean::booleanValue ( ) const [inline]

#### Returns

the primitive boolean value of this object

### 6.138.2.2 virtual int decaf::lang::Boolean::compareTo ( const Boolean & b ) const [virtual]

Compares this **Boolean** (p. 856) instance with another.

#### Parameters

<i>b</i>	- the <b>Boolean</b> (p. 856) instance to be compared
----------	---

#### Returns

zero if this object represents the same boolean value as the argument; a positive value if this object represents true and the argument represents false; and a negative value if this object represents false and the argument represents true

Implements **decaf::lang::Comparable**< **Boolean** > (p. 1249).

**6.138.2.3** `virtual int decaf::lang::Boolean::compareTo ( const bool & b ) const` [virtual]

Compares this **Boolean** (p. 856) instance with another.

#### Parameters

<i>b</i>	- the <b>Boolean</b> (p. 856) instance to be compared
----------	---

#### Returns

zero if this object represents the same boolean value as the argument; a positive value if this object represents true and the argument represents false; and a negative value if this object represents false and the argument represents true

Implements **decaf::lang::Comparable**< **bool** > (p. 1249).

**6.138.2.4** `bool decaf::lang::Boolean::equals ( const bool & b ) const` [inline, virtual]

#### Returns

true if the two **Boolean** (p. 856) Objects have the same value.

Implements **decaf::lang::Comparable**< **bool** > (p. 1250).

**6.138.2.5** `bool decaf::lang::Boolean::equals ( const Boolean & b ) const` [inline, virtual]

#### Returns

true if the two **Boolean** (p. 856) Objects have the same value.

Implements **decaf::lang::Comparable**< **Boolean** > (p. 1250).

**6.138.2.6** `virtual bool decaf::lang::Boolean::operator< ( const bool & value ) const` [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

**Returns**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **bool** > (p. 1250).

**6.138.2.7** `virtual bool decaf::lang::Boolean::operator< ( const Boolean & value ) const`  
[virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

**Parameters**

<i>value</i>	- the value to be compared to this one.
--------------	---

**Returns**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Boolean** > (p. 1250).

**6.138.2.8** `virtual bool decaf::lang::Boolean::operator== ( const bool & value ) const`  
[virtual]

Compares equality between this object and the one passed.

**Parameters**

<i>value</i>	- the value to be compared to this one.
--------------	---

**Returns**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **bool** > (p. 1250).

**6.138.2.9** `virtual bool decaf::lang::Boolean::operator== ( const Boolean & value ) const`  
[virtual]

Compares equality between this object and the one passed.

**Parameters**

<i>value</i>	- the value to be compared to this one.
--------------	---

**Returns**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Boolean** > (p. 1250).

**6.138.2.10** `static bool decaf::lang::Boolean::parseBoolean ( const std::string & value )`  
[static]

Parses the **String** (p. 3775) passed and extracts an bool.

#### Parameters

<i>value</i>	The std::string value to parse
--------------	--------------------------------

#### Returns

bool value

**6.138.2.11** `static std::string decaf::lang::Boolean::toString ( bool value )` [static]

Converts the bool to a **String** (p. 3775) representation.

#### Parameters

<i>value</i>	The bool value to convert.
--------------	----------------------------

#### Returns

std::string representation of the bool value passed.

**6.138.2.12** `std::string decaf::lang::Boolean::toString ( ) const`

#### Returns

the string representation of this Booleans value.

**6.138.2.13** `static Boolean decaf::lang::Boolean::valueOf ( bool value )` [static]

#### Parameters

<i>value</i>	The bool value to convert to a <b>Boolean</b> (p. 856) instance.
--------------	--

#### Returns

a **Boolean** (p. 856) instance of the primitive boolean value

**6.138.2.14** `static Boolean decaf::lang::Boolean::valueOf ( const std::string & value )`  
`[static]`

#### Parameters

<i>value</i>	The std::string value to convert to a <b>Boolean</b> (p. 856) instance.
--------------	---

#### Returns

a **Boolean** (p. 856) instance of the string value

### 6.138.3 Field Documentation

**6.138.3.1** `const Boolean decaf::lang::Boolean::_FALSE` `[static]`

The Class object representing the primitive false boolean.

**6.138.3.2** `const Boolean decaf::lang::Boolean::_TRUE` `[static]`

The Class object representing the primitive type boolean.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Boolean.h`

## 6.139 activemq::commands::BooleanExpression Class Reference

```
#include <src/main/activemq/commands/BooleanExpression.h>
```

Inheritance diagram for activemq::commands::BooleanExpression:

### Public Member Functions

- **BooleanExpression** ()
- virtual **~BooleanExpression** ()
- virtual **DataStructure \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src AMQCPP\_UNUSED)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

- virtual bool **equals** (const **DataStructure** \*value) const

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

### 6.139.1 Constructor & Destructor Documentation

6.139.1.1 **activemq::commands::BooleanExpression::BooleanExpression** ( ) [inline]

6.139.1.2 **virtual activemq::commands::BooleanExpression::~~BooleanExpression** ( )  
[inline, virtual]

### 6.139.2 Member Function Documentation

6.139.2.1 **virtual DataStructure\* activemq::commands::BooleanExpression::cloneDataStructure**  
( ) const [inline, virtual]

Clone this obbjct and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

6.139.2.2 **virtual void activemq::commands::BooleanExpression::copyDataStructure** ( const  
**DataStructure \*src** *AMQCPP\_UNUSED* ) [inline, virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from **activemq::commands::BaseDataStructure** (p. 839).

6.139.2.3 **virtual bool activemq::commands::BooleanExpression::equals** ( const  
**DataStructure \* value** ) const [inline, virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

## 6.140 `activemq::wireformat::openwire::utils::BooleanStream` Class Reference 863

### Returns

true if `DataStream`'s are Equal.

Implements `activemq::commands::DataStream` (p. 1716).

References `activemq::commands::BaseDataStream::equals()`.

**6.139.2.4** `virtual std::string activemq::commands::BooleanExpression::toString ( ) const`  
[inline, virtual]

Returns a string containing the information for this `DataStream` (p. 1713) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStream` (p. 841).

References `activemq::commands::BaseDataStream::toString()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BooleanExpression.h`

## 6.140 `activemq::wireformat::openwire::utils::BooleanStream` Class Reference

Manages the writing and reading of boolean data streams to and from a data source such as a `DataInputStream` or `DataOutputStream`.

```
#include <src/main/activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Public Member Functions

- `BooleanStream ()`
- `virtual ~BooleanStream ()`
- `bool readBoolean () throw ( decaf::io::IOException )`  
*Read a boolean data element from the internal data buffer.*
- `void writeBoolean (bool value) throw ( decaf::io::IOException )`  
*Writes a Boolean value to the internal data buffer.*
- `void marshal (decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Marshal the data to a DataOutputStream.*

- void **marshal** (std::vector< unsigned char > &dataOut)  
*Marshal the data to a STL vector of unsigned chars.*
- void **unmarshal** (decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )  
*Unmarshal a Boolean data stream from the Input Stream.*
- void **clear** ()  
*Clears to old position markers, data starts at the beginning.*
- int **marshalledSize** ()  
*Calc the size that data is marshalled to.*

### 6.140.1 Detailed Description

Manages the writing and reading of boolean data streams to and from a data source such as a DataInputStream or DataOutputStream. The booleans are stored as single bits in the stream, with the stream size pre-pended to the stream when the data is marshalled.

The serialized form of the size field can be between 1 and 3 bytes. If the number of used bytes < 64, size=1 byte If the number of used bytes >=64 and < 256 (size of an unsigned byte), size=2 bytes If the number of used bytes >=256, size=3 bytes

The high-order 2 bits (128 and 64) of the first byte of the size field are used to encode the information about the number of bytes in the size field. The only time the first byte will contain a value >=64 is if there are more bytes in the size field. If the first byte < 64, the value of the byte is simply the size value. If the first byte = 0xC0, the following unsigned byte is the size field. If the first byte = 0x80, the following short (two bytes) are the size field.

### 6.140.2 Constructor & Destructor Documentation

6.140.2.1 **activemq::wireformat::openwire::utils::BooleanStream::BooleanStream ( )**

6.140.2.2 **virtual activemq::wireformat::openwire::utils::BooleanStream::~~BooleanStream ( )**  
[inline, virtual]

### 6.140.3 Member Function Documentation

6.140.3.1 **void activemq::wireformat::openwire::utils::BooleanStream::clear ( )**

Clears to old position markers, data starts at the beginning.

6.140.3.2 **void activemq::wireformat::openwire::utils::BooleanStream::marshal ( std::vector< unsigned char > & dataOut )**

Marshal the data to a STL vector of unsigned chars.



## 6.140 activemq::wireformat::openwire::utils::BooleanStream Class Reference 865

### Parameters

<i>dataOut</i>	- reference to a vector to write the data to.
----------------	---

**6.140.3.3** void activemq::wireformat::openwire::utils::BooleanStream::marshal ( decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException )

Marshal the data to a DataOutputStream.

### Parameters

<i>dataOut</i>	- Stream to write the data to.
----------------	--------------------------------

**6.140.3.4** int activemq::wireformat::openwire::utils::BooleanStream::marshalledSize ( )

Calc the size that data is marshalled to.

### Returns

int size of marshalled data.

**6.140.3.5** bool activemq::wireformat::openwire::utils::BooleanStream::readBoolean ( ) throw ( decaf::io::IOException )

Read a boolean data element from the internal data buffer.

### Returns

boolean from the stream

**6.140.3.6** void activemq::wireformat::openwire::utils::BooleanStream::unmarshal ( decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException )

Unmarshal a Boolean data stream from the Input Stream.

### Parameters

<i>dataIn</i>	- Input Stream to read data from.
---------------	-----------------------------------

**6.140.3.7** void activemq::wireformat::openwire::utils::BooleanStream::writeBoolean ( bool *value* ) throw ( decaf::io::IOException )

Writes a Boolean value to the internal data buffer.

**Parameters**

<i>value</i>	- boolean data to write.
--------------	--------------------------

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/utis/**BooleanStream.h**

## 6.141 decaf::util::concurrent::BrokenBarrierException Class Reference

```
#include <src/main/decaf/util/concurrent/BrokenBarrierException.h>
```

Inheritance diagram for decaf::util::concurrent::BrokenBarrierException:

**Public Member Functions**

- **BrokenBarrierException** () throw ()  
*Default Constructor.*
- **BrokenBarrierException** (const decaf::lang::Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **BrokenBarrierException** (const **BrokenBarrierException** &ex) throw ()  
*Copy Constructor.*
- **BrokenBarrierException** (const std::exception \*cause) throw ()  
*Constructor.*
- **BrokenBarrierException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **BrokenBarrierException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **BrokenBarrierException** \* clone () const  
*Clones this exception.*
- virtual ~**BrokenBarrierException** () throw ()

### 6.141.1 Constructor & Destructor Documentation

**6.141.1.1** `decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException ( ) throw ()` `[inline]`

Default Constructor.

**6.141.1.2** `decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException ( const decaf::lang::Exception & ex ) throw ()` `[inline]`

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.141.1.3** `decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException ( const BrokenBarrierException & ex ) throw ()` `[inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	The Exception to copy in this new instance.
-----------	---

**6.141.1.4** `decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException ( const std::exception * cause ) throw ()` `[inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.141.1.5** `decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.
<i>msg</i>	- The message to report

...	- list of primitives that are formatted into the message
-----	--

**6.141.1.6** `decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.
<i>cause</i>	- The exception that was the cause for this one to be thrown.
<i>msg</i>	- The message to report
...	- list of primitives that are formatted into the message

**6.141.1.7** `virtual decaf::util::concurrent::BrokenBarrierException::~~BrokenBarrierException ( ) throw () [inline, virtual]`

## 6.141.2 Member Function Documentation

**6.141.2.1** `virtual BrokenBarrierException* decaf::util::concurrent::BrokenBarrierException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

a new instance of an exception that is a clone of this one.

Reimplemented from `decaf::lang::Exception` (p. 1889).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/BrokenBarrierException.h`

## 6.142 activemq::commands::BrokerError Class Reference

This class represents an Exception sent from the Broker.

```
#include <src/main/activemq/commands/BrokerError.h>
```

Inheritance diagram for activemq::commands::BrokerError:

## Data Structures

- struct **StackTraceElement**

## Public Member Functions

- **BrokerError** ()
- virtual **~BrokerError** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the **DataStructure** (p. 1713) Type as defined in CommandTypes.h.*
- virtual **BrokerError** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual **decaf::lang::Pointer**< **commands::Command** > **visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*
- virtual const std::string & **getMessage** () const  
*Gets the string holding the error message.*
- virtual void **setMessage** (const std::string &message)  
*Sets the string that contains the error **Message** (p. 2596).*
- virtual const std::string & **getExceptionClass** () const  
*Gets the string holding the Exception Class name.*
- virtual void **setExceptionClass** (const std::string &exceptionClass)  
*Sets the string that contains the Exception Class name.*
- virtual const **decaf::lang::Pointer**< **BrokerError** > & **getCause** () const  
*Gets the Broker Error that caused this exception.*
- virtual void **setCause** (const **decaf::lang::Pointer**< **BrokerError** > &cause)  
*Sets the Broker Error that caused this exception.*

- virtual const std::vector< **decaf::lang::Pointer**< **StackTraceElement** > > &**getStackTraceElements** () const  
*Gets the Stack Trace Elements for the Exception.*
- virtual void **setStackTraceElements** (const std::vector< **decaf::lang::Pointer**< **StackTraceElement** > > &stackTraceElements)  
*Sets the Stack Trace Elements for this Exception.*

### 6.142.1 Detailed Description

This class represents an Exception sent from the Broker. The Broker sends java Throwables, so we must mimic its structure here.

### 6.142.2 Constructor & Destructor Documentation

6.142.2.1 **activemq::commands::BrokerError::BrokerError** ( )

6.142.2.2 **virtual activemq::commands::BrokerError::~~BrokerError** ( ) [virtual]

### 6.142.3 Member Function Documentation

6.142.3.1 **virtual BrokerError\* activemq::commands::BrokerError::cloneDataStructure** ( )  
**const** [inline, virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

References copyDataStructure().

6.142.3.2 **virtual void activemq::commands::BrokerError::copyDataStructure** ( **const DataStructure \* src** ) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

Referenced by cloneDataStructure().

**6.142.3.3** `virtual const decaf::lang::Pointer<BrokerError>&  
activemq::commands::BrokerError::getCause ( ) const [inline,  
virtual]`

Gets the Broker Error that caused this exception.

#### Returns

Broker Error Pointer

**6.142.3.4** `virtual unsigned char activemq::commands::BrokerError::getDataStructureType ( )  
const [inline, virtual]`

Get the **DataStructure** (p. 1713) Type as defined in CommandTypes.h.

#### Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1717).

**6.142.3.5** `virtual const std::string& activemq::commands::BrokerError::getExceptionClass ( )  
const [inline, virtual]`

Gets the string holding the Exception Class name.

#### Returns

Exception Class name

**6.142.3.6** `virtual const std::string& activemq::commands::BrokerError::getMessage ( ) const  
[inline, virtual]`

Gets the string holding the error message.

#### Returns

String **Message** (p. 2596)

**6.142.3.7** `virtual const std::vector< decaf::lang::Pointer<StackTraceElement> >&  
activemq::commands::BrokerError::getStackTraceElements ( ) const [inline,  
virtual]`

Gets the Stack Trace Elements for the Exception.

#### Returns

Stack Trace Elements

**6.142.3.8** `virtual void activemq::commands::BrokerError::setCause ( const  
decaf::lang::Pointer< BrokerError > & cause ) [inline,  
virtual]`

Sets the Broker Error that caused this exception.

#### Parameters

<i>cause</i>	- Broker Error
--------------	----------------

**6.142.3.9** `virtual void activemq::commands::BrokerError::setExceptionClass ( const std::string  
& exceptionClass ) [inline, virtual]`

Sets the string that contains the Exception Class name.

#### Parameters

<i>exception- Class</i>	- String Exception Class name
-----------------------------	-------------------------------

**6.142.3.10** `virtual void activemq::commands::BrokerError::setMessage ( const std::string &  
message ) [inline, virtual]`

Sets the string that contains the error **Message** (p. 2596).

#### Parameters

<i>message</i>	- String Error <b>Message</b> (p. 2596)
----------------	---

**6.142.3.11** `virtual void activemq::commands::BrokerError::setStackTraceElements ( const  
std::vector< decaf::lang::Pointer< StackTraceElement > > &  
stackTraceElements ) [inline, virtual]`

Sets the Stack Trace Elements for this Exception.

#### Parameters

<i>stack- TraceEle- ments</i>	- Stack Trace Elements
---------------------------------------	------------------------



```
6.142.3.12 virtual decaf::lang::Pointer<commands::Command>
      activemq::commands::BrokerError::visit ( activemq::state::CommandVisitor
      * visitor ) throw ( exceptions::ActiveMQException ) [virtual]
```

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

### Returns

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**BrokerError.h**

## 6.143 activemq::exceptions::BrokerException Class Reference

```
#include <src/main/activemq/exceptions/BrokerException.h>
```

Inheritance diagram for activemq::exceptions::BrokerException:

### Public Member Functions

- **BrokerException** () throw ()
- **BrokerException** (const exceptions::ActiveMQException &ex) throw ()
- **BrokerException** (const **BrokerException** &ex) throw ()
- **BrokerException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()
- **BrokerException** (const char \*file, const int lineNumber, const **commands::BrokerError** \*error) throw ()
- virtual **BrokerException** \* **clone** () const  
*Clones this exception.*
- virtual ~**BrokerException** () throw ()

### 6.143.1 Constructor & Destructor Documentation

6.143.1.1 `activemq::exceptions::BrokerException::BrokerException ( ) throw ()` `[inline]`

6.143.1.2 `activemq::exceptions::BrokerException::BrokerException ( const exceptions::ActiveMQException & ex ) throw ()` `[inline]`

6.143.1.3 `activemq::exceptions::BrokerException::BrokerException ( const BrokerException & ex ) throw ()` `[inline]`

6.143.1.4 `activemq::exceptions::BrokerException::BrokerException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()` `[inline]`

6.143.1.5 `activemq::exceptions::BrokerException::BrokerException ( const char * file, const int lineNumber, const commands::BrokerError * error ) throw ()` `[inline]`

References `decaf::lang::Exception::getMessage()`.

6.143.1.6 `virtual activemq::exceptions::BrokerException::~~BrokerException ( ) throw ()`  
`[inline, virtual]`

### 6.143.2 Member Function Documentation

6.143.2.1 `virtual BrokerException* activemq::exceptions::BrokerException::clone ( ) const`  
`[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from `activemq::exceptions::ActiveMQException` (p. 349).

The documentation for this class was generated from the following file:

- `src/main/activemq/exceptions/BrokerException.h`

## 6.144 `activemq::commands::BrokerId` Class Reference

```
#include <src/main/activemq/commands/BrokerId.h>
```

Inheritance diagram for `activemq::commands::BrokerId`:

### Public Types

- `typedef decaf::lang::PointerComparator< BrokerId > COMPARATOR`

## Public Member Functions

- **BrokerId** ()
- **BrokerId** (const **BrokerId** &other)
- virtual ~**BrokerId** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **BrokerId** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getValue** () const
- virtual std::string & **getValue** ()
- virtual void **setValue** (const std::string &value)
- virtual int **compareTo** (const **BrokerId** &value) const
- virtual bool **equals** (const **BrokerId** &value) const
- virtual bool **operator==** (const **BrokerId** &value) const
- virtual bool **operator<** (const **BrokerId** &value) const
- **BrokerId** & **operator=** (const **BrokerId** &other)

## Static Public Attributes

- static const unsigned char **ID\_BROKERID** = 124

## Protected Attributes

- std::string **value**

### 6.144.1 Member Typedef Documentation

6.144.1.1 `typedef decaf::lang::PointerComparator<BrokerId>  
activemq::commands::BrokerId::COMPARATOR`

### 6.144.2 Constructor & Destructor Documentation

6.144.2.1 `activemq::commands::BrokerId::BrokerId ( )`

6.144.2.2 `activemq::commands::BrokerId::BrokerId ( const BrokerId & other )`

6.144.2.3 `virtual activemq::commands::BrokerId::~~BrokerId ( ) [virtual]`

### 6.144.3 Member Function Documentation

6.144.3.1 `virtual BrokerId* activemq::commands::BrokerId::cloneDataStructure ( ) const  
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1714).

6.144.3.2 `virtual int activemq::commands::BrokerId::compareTo ( const BrokerId & value )  
const [virtual]`

6.144.3.3 `virtual void activemq::commands::BrokerId::copyDataStructure ( const  
DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Implements `activemq::commands::DataStructure` (p. 1715).

6.144.3.4 `virtual bool activemq::commands::BrokerId::equals ( const DataStructure * value  
)const [virtual]`

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1716).

**6.144.3.5** `virtual bool activemq::commands::BrokerId::equals ( const BrokerId & value ) const [virtual]`

**6.144.3.6** `virtual unsigned char activemq::commands::BrokerId::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

**6.144.3.7** `virtual const std::string& activemq::commands::BrokerId::getValue ( ) const [virtual]`

**6.144.3.8** `virtual std::string& activemq::commands::BrokerId::getValue ( ) [virtual]`

**6.144.3.9** `virtual bool activemq::commands::BrokerId::operator< ( const BrokerId & value ) const [virtual]`

**6.144.3.10** `BrokerId& activemq::commands::BrokerId::operator= ( const BrokerId & other )`

**6.144.3.11** `virtual bool activemq::commands::BrokerId::operator== ( const BrokerId & value ) const [virtual]`

**6.144.3.12** `virtual void activemq::commands::BrokerId::setValue ( const std::string & value ) [virtual]`

**6.144.3.13** `virtual std::string activemq::commands::BrokerId::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 841).

### 6.144.4 Field Documentation

6.144.4.1 `const unsigned char activemq::commands::BrokerId::ID_BROKERID = 124` `[static]`

6.144.4.2 `std::string activemq::commands::BrokerId::value` `[protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BrokerId.h`

## 6.145 `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller` Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 878).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdMarshal
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller`:

### Public Member Functions

- **BrokerIdMarshaller** ()
- virtual `~BrokerIdMarshaller` ()
- virtual `commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataInputStream *dataIn`, `utils::BooleanStream *bs`)  
`throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `utils::BooleanStream *bs`)  
`throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataOutputStream *dataOut`, `utils::BooleanStream *bs`)  
`throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.145.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 878). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.145.2 Constructor & Destructor Documentation

**6.145.2.1** **activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::BrokerIdMarshaller**  
( ) [inline]

**6.145.2.2** **virtual activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::~~BrokerIdMarshaller**  
( ) [inline, virtual]

### 6.145.3 Member Function Documentation

**6.145.3.1** **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.145.3.2** **virtual unsigned char activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

```
6.145.3.3  virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1675).

```
6.145.3.4  virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1683).

```
6.145.3.5  virtual int activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.



## 6.145 activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller Class Reference 881

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.145.3.6 virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.145.3.7 virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.

<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**BrokerIdMarshaller.h**

## 6.146 activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 882).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/BrokerIdMarshal
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller**:

### Public Member Functions

- **BrokerIdMarshaller** ()
- virtual  $\sim$ **BrokerIdMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.146.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 882). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.146.2 Constructor & Destructor Documentation

6.146.2.1 **activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::BrokerIdMarshaller**  
( ) [inline]

6.146.2.2 **virtual activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::~~BrokerIdMarshaller**  
( ) [inline, virtual]

### 6.146.3 Member Function Documentation

6.146.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.146.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.146.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.146.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

## 6.146 activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller Class Reference 885

**6.146.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1690).

**6.146.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1698).

```
6.146.3.7 virtual void activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**BrokerIdMarshaller.h**

## 6.147 activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 886).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller**:

#### Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.147.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 886). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.147.2 Constructor & Destructor Documentation

**6.147.2.1** **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::BrokerIdMarshaller**  
( ) [inline]

**6.147.2.2** **virtual activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::~~BrokerIdMarshaller**  
( ) [inline, virtual]

### 6.147.3 Member Function Documentation

**6.147.3.1** **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.147.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.147.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.147.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source



## 6.147 activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller Class Reference 889

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.147.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.147.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.147.3.7 virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**BrokerIdMarshaller.h**

## 6.148 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 890).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller**:

#### Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.148.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 890). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.148.2 Constructor & Destructor Documentation

- 6.148.2.1 **activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::BrokerIdMarshaller**  
( ) [inline]
- 6.148.2.2 **virtual activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::~~BrokerIdMarshaller**  
( ) [inline, virtual]

### 6.148.3 Member Function Documentation

- 6.148.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.148.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.148.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.148.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## 6.148 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller Class Reference 893

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.148.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.148.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.148.3.7 virtual void activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
  * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**BrokerIdMarshaller.h**

## 6.149 activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 894).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/BrokerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller**:

#### Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.149.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p.894). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.149.2 Constructor & Destructor Documentation

- 6.149.2.1 **activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::BrokerIdMarshaller**  
( ) [inline]
- 6.149.2.2 **virtual activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::~~BrokerIdMarshaller**  
( ) [inline, virtual]

### 6.149.3 Member Function Documentation

- 6.149.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.149.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.149.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.149.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source



## 6.149 activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller Class Reference 897

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.149.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.149.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.149.3.7 virtual void activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
  * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**BrokerIdMarshaller.h**

## 6.150 activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 898).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller**:

#### Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
  
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
  
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
  
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
  
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.150.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p.898). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.150.2 Constructor & Destructor Documentation

- 6.150.2.1 **activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::BrokerIdMarshaller**  
( ) [inline]
  
- 6.150.2.2 **virtual activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::~~BrokerIdMarshaller**  
( ) [inline, virtual]

### 6.150.3 Member Function Documentation

- 6.150.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.150.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.150.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.150.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## 6.150 activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller Class Reference 901

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

```
6.150.3.5 virtual int activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.150.3.6 virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.150.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdMarshaller.h`

## 6.151 activemq::commands::BrokerInfo Class Reference

```
#include <src/main/activemq/commands/BrokerInfo.h>
```

Inheritance diagram for `activemq::commands::BrokerInfo`:

#### Public Member Functions

- **BrokerInfo** ()
- virtual **~BrokerInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **BrokerInfo** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)

*Copy the contents of the passed object into this object's members, overwriting any existing data.*

- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **BrokerId** > & **getBrokerId** () const
- virtual **Pointer**< **BrokerId** > & **getBrokerId** ()
- virtual void **setBrokerId** (const **Pointer**< **BrokerId** > &brokerId)
- virtual const std::string & **getBrokerURL** () const
- virtual std::string & **getBrokerURL** ()
- virtual void **setBrokerURL** (const std::string &brokerURL)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > & **getPeerBrokerInfos** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > & **getPeerBrokerInfos** ()
- virtual void **setPeerBrokerInfos** (const std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > &peerBrokerInfos)
- virtual const std::string & **getBrokerName** () const
- virtual std::string & **getBrokerName** ()
- virtual void **setBrokerName** (const std::string &brokerName)
- virtual bool **isSlaveBroker** () const
- virtual void **setSlaveBroker** (bool slaveBroker)
- virtual bool **isMasterBroker** () const
- virtual void **setMasterBroker** (bool masterBroker)
- virtual bool **isFaultTolerantConfiguration** () const
- virtual void **setFaultTolerantConfiguration** (bool faultTolerantConfiguration)
- virtual bool **isDuplexConnection** () const
- virtual void **setDuplexConnection** (bool duplexConnection)
- virtual bool **isNetworkConnection** () const
- virtual void **setNetworkConnection** (bool networkConnection)
- virtual long long **getConnectionId** () const
- virtual void **setConnectionId** (long long connectionId)
- virtual const std::string & **getBrokerUploadUrl** () const
- virtual std::string & **getBrokerUploadUrl** ()
- virtual void **setBrokerUploadUrl** (const std::string &brokerUploadUrl)
- virtual const std::string & **getNetworkProperties** () const
- virtual std::string & **getNetworkProperties** ()
- virtual void **setNetworkProperties** (const std::string &networkProperties)
- virtual bool **isBrokerInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_BROKERINFO** = 2

### Protected Attributes

- **Pointer< BrokerId > brokerId**
- std::string **brokerURL**
- std::vector< decaf::lang::Pointer< BrokerInfo > > **peerBrokerInfos**
- std::string **brokerName**
- bool **slaveBroker**
- bool **masterBroker**
- bool **faultTolerantConfiguration**
- bool **duplexConnection**
- bool **networkConnection**
- long long **connectionId**
- std::string **brokerUploadUrl**
- std::string **networkProperties**

## 6.151.1 Constructor & Destructor Documentation

6.151.1.1 **activemq::commands::BrokerInfo::BrokerInfo ( )**

6.151.1.2 **virtual activemq::commands::BrokerInfo::~~BrokerInfo ( )** [virtual]

## 6.151.2 Member Function Documentation

6.151.2.1 **virtual BrokerInfo\* activemq::commands::BrokerInfo::cloneDataStructure ( )**  
**const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

6.151.2.2 **virtual void activemq::commands::BrokerInfo::copyDataStructure ( const DataStructure \* src )** [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters



<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

**6.151.2.3** `virtual bool activemq::commands::BrokerInfo::equals ( const DataStructure *  
value ) const` [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

- 6.151.2.4 `virtual const Pointer<BrokerId>& activemq::commands::BrokerInfo::getBrokerId ( ) const [virtual]`
- 6.151.2.5 `virtual Pointer<BrokerId>& activemq::commands::BrokerInfo::getBrokerId ( ) [virtual]`
- 6.151.2.6 `virtual std::string& activemq::commands::BrokerInfo::getBrokerName ( ) [virtual]`
- 6.151.2.7 `virtual const std::string& activemq::commands::BrokerInfo::getBrokerName ( ) const [virtual]`
- 6.151.2.8 `virtual const std::string& activemq::commands::BrokerInfo::getBrokerUploadUrl ( ) const [virtual]`
- 6.151.2.9 `virtual std::string& activemq::commands::BrokerInfo::getBrokerUploadUrl ( ) [virtual]`
- 6.151.2.10 `virtual const std::string& activemq::commands::BrokerInfo::getBrokerURL ( ) const [virtual]`
- 6.151.2.11 `virtual std::string& activemq::commands::BrokerInfo::getBrokerURL ( ) [virtual]`
- 6.151.2.12 `virtual long long activemq::commands::BrokerInfo::getConnectionId ( ) const [virtual]`
- 6.151.2.13 `virtual unsigned char activemq::commands::BrokerInfo::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

### Returns

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

6.151.2.14 `virtual const std::string& activemq::commands::BrokerInfo::getNetworkProperties ( ) const [virtual]`

6.151.2.15 `virtual std::string& activemq::commands::BrokerInfo::getNetworkProperties ( ) [virtual]`

6.151.2.16 `virtual const std::vector< decaf::lang::Pointer<BrokerInfo> >& activemq::commands::BrokerInfo::getPeerBrokerInfos ( ) const [virtual]`

6.151.2.17 `virtual std::vector< decaf::lang::Pointer<BrokerInfo> >& activemq::commands::BrokerInfo::getPeerBrokerInfos ( ) [virtual]`

6.151.2.18 `virtual bool activemq::commands::BrokerInfo::isBrokerInfo ( ) const [inline, virtual]`

### Returns

an answer of true to the `isBrokerInfo()` (p. 907) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 767).

- 
- 6.151.2.19 `virtual bool activemq::commands::BrokerInfo::isDuplexConnection ( ) const`  
[virtual]
  - 6.151.2.20 `virtual bool activemq::commands::BrokerInfo::isFaultTolerantConfiguration ( )`  
`const` [virtual]
  - 6.151.2.21 `virtual bool activemq::commands::BrokerInfo::isMasterBroker ( ) const`  
[virtual]
  - 6.151.2.22 `virtual bool activemq::commands::BrokerInfo::isNetworkConnection ( ) const`  
[virtual]
  - 6.151.2.23 `virtual bool activemq::commands::BrokerInfo::isSlaveBroker ( ) const`  
[virtual]
  - 6.151.2.24 `virtual void activemq::commands::BrokerInfo::setBrokerId ( const Pointer<`  
`BrokerId > &brokerId )` [virtual]
  - 6.151.2.25 `virtual void activemq::commands::BrokerInfo::setBrokerName ( const std::string &`  
`brokerName )` [virtual]
  - 6.151.2.26 `virtual void activemq::commands::BrokerInfo::setBrokerUploadUrl ( const std::string`  
`& brokerUploadUrl )` [virtual]
  - 6.151.2.27 `virtual void activemq::commands::BrokerInfo::setBrokerURL ( const std::string &`  
`brokerURL )` [virtual]
  - 6.151.2.28 `virtual void activemq::commands::BrokerInfo::setConnectionId ( long long`  
`connectionId )` [virtual]
  - 6.151.2.29 `virtual void activemq::commands::BrokerInfo::setDuplexConnection ( bool`  
`duplexConnection )` [virtual]
  - 6.151.2.30 `virtual void activemq::commands::BrokerInfo::setFaultTolerantConfiguration ( bool`  
`faultTolerantConfiguration )` [virtual]
  - 6.151.2.31 `virtual void activemq::commands::BrokerInfo::setMasterBroker ( bool masterBroker`  
`)` [virtual]
  - 6.151.2.32 `virtual void activemq::commands::BrokerInfo::setNetworkConnection ( bool`  
`networkConnection )` [virtual]
  - 6.151.2.33 `virtual void activemq::commands::BrokerInfo::setNetworkProperties ( const`  
`std::string & networkProperties )` [virtual]
  - 6.151.2.34 `virtual void activemq::commands::BrokerInfo::setPeerBrokerInfos ( const`  
`std::vector< decaf::lang::Pointer< BrokerInfo > > & peerBrokerInfos )`  
[virtual]
  - 6.151.2.35 `virtual void activemq::commands::BrokerInfo::setSlaveBroker ( bool slaveBroker )`  
[virtual]
- 
- Generated on Tue Mar 8 2011 17:10:05 for activemq-cpp-3.2.5 by Doxygen
- 6.151.2.36 `virtual std::string activemq::commands::BrokerInfo::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

**Returns**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

```
6.151.2.37 virtual Pointer<Command> activemq::commands::BrokerInfo::visit  
    ( activemq::state::CommandVisitor * visitor ) throw (   
        exceptions::ActiveMQException ) [virtual]
```

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

**Returns**

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

### 6.151.3 Field Documentation

- 6.151.3.1 **Pointer<BrokerId> activemq::commands::BrokerInfo::brokerId**  
[protected]
- 6.151.3.2 **std::string activemq::commands::BrokerInfo::brokerName**  
[protected]
- 6.151.3.3 **std::string activemq::commands::BrokerInfo::brokerUploadUrl**  
[protected]
- 6.151.3.4 **std::string activemq::commands::BrokerInfo::brokerURL**  
[protected]
- 6.151.3.5 **long long activemq::commands::BrokerInfo::connectionId**  
[protected]
- 6.151.3.6 **bool activemq::commands::BrokerInfo::duplexConnection**  
[protected]
- 6.151.3.7 **bool activemq::commands::BrokerInfo::faultTolerantConfiguration**  
[protected]
- 6.151.3.8 **const unsigned char activemq::commands::BrokerInfo::ID\_-  
BROKERINFO = 2** [static]
- 6.151.3.9 **bool activemq::commands::BrokerInfo::masterBroker** [protected]
- 6.151.3.10 **bool activemq::commands::BrokerInfo::networkConnection**  
[protected]
- 6.151.3.11 **std::string activemq::commands::BrokerInfo::networkProperties**  
[protected]
- 6.151.3.12 **std::vector< decaf::lang::Pointer<BrokerInfo> >  
activemq::commands::BrokerInfo::peerBrokerInfos** [protected]
- 6.151.3.13 **bool activemq::commands::BrokerInfo::slaveBroker** [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**BrokerInfo.h**

## 6.152 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 910).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller:

## Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.152.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p.910). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.152.2 Constructor & Destructor Documentation

6.152.2.1 `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::BrokerInfoMarshaller ( ) [inline]`

6.152.2.2 `virtual activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::~~BrokerInfoMarshaller ( ) [inline, virtual]`

## 6.152.3 Member Function Documentation

6.152.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.152.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.152.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 772).

**6.152.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 774).

**6.152.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 775).

```
6.152.3.6 virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 776).

```
6.152.3.7 virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 777).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfoMarshaller.h`

## 6.153 **activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 915).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller`:

### Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.153.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 915). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.153.2 Constructor & Destructor Documentation

6.153.2.1 `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::BrokerInfoMarshaller ( ) [inline]`

6.153.2.2 `virtual activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::~~BrokerInfoMarshaller ( ) [inline, virtual]`

### 6.153.3 Member Function Documentation

6.153.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.153.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.153.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 780).

**6.153.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 781).

**6.153.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 782).

```
6.153.3.6 virtual void activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 783).

```
6.153.3.7 virtual void activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfoMarshaller.h`

## 6.154 **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 919).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller**:

## Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.154.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 919). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.154.2 Constructor & Destructor Documentation

6.154.2.1 **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::BrokerInfoMarshaller**  
 ( ) [inline]

6.154.2.2 **virtual activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::~~BrokerInfoMarshaller**  
 ( ) [inline, virtual]

### 6.154.3 Member Function Documentation

6.154.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::createObject** ( )  
 const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.154.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value



Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.154.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 787).

**6.154.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 788).

```
6.154.3.5 virtual int activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 789).

```
6.154.3.6 virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 790).

## 6.155 activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller

### Class Reference

923

6.154.3.7 virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 792).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**BrokerInfoMarshaller.h**

## 6.155 activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 923).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller:

#### Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.155.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 923). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.155.2 Constructor & Destructor Documentation

6.155.2.1 **activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::BrokerInfoMarshaller**  
( ) [inline]

6.155.2.2 **virtual activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::~~BrokerInfoMarshaller**  
( ) [inline, virtual]

### 6.155.3 Member Function Documentation

6.155.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.155.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaller.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.155.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 794).

**6.155.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 795).

```
6.155.3.5  virtual int activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 796).

```
6.155.3.6  virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

## 6.156 activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller

### Class Reference

927

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 797).

**6.155.3.7** virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**BrokerInfoMarshaller.h**

## 6.156 activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 927).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/BrokerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller:

## Public Member Functions

- **BrokerInfoMarshaller ()**
- virtual **~BrokerInfoMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn)** throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.156.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 927). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.156.2 Constructor & Destructor Documentation

- 6.156.2.1 `activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::BrokerInfoMarshaller ( ) [inline]`
- 6.156.2.2 `virtual activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::~~BrokerInfoMarshaller ( ) [inline, virtual]`

## 6.156.3 Member Function Documentation

- 6.156.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

- 6.156.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

- 6.156.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 801).

**6.156.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 802).

**6.156.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 803).

**6.156.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 804).

**6.156.3.7** `virtual void activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 806).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/BrokerInfoMarshaller.h`

## 6.157 `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 932).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfoMarsh
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller`:

### Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.157.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p.932). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.157.2 Constructor & Destructor Documentation

**6.157.2.1** `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::BrokerInfoMarshaller ( ) [inline]`

**6.157.2.2** `virtual activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::~~BrokerInfoMarshaller ( ) [inline, virtual]`

### 6.157.3 Member Function Documentation

**6.157.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

**6.157.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.157.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 808).

**6.157.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 809).

**6.157.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 810).

**6.157.3.6** virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::tightMarshal2  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure  
 \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
 utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
 [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 811).

**6.157.3.7** virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::tightUnmarshal  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream  
 \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 813).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**BrokerInfoMarshaller.h**

**6.158 decaf::nio::Buffer Class Reference**

A container for data of a specific primitive type.

```
#include <src/main/decaf/nio/Buffer.h>
```

Inheritance diagram for decaf::nio::Buffer:

**Public Member Functions**

- **Buffer** (int capacity)
- **Buffer** (const **Buffer** &other)
- virtual ~**Buffer** ()
- virtual int **capacity** () const
- virtual int **position** () const
- virtual **Buffer** & **position** (int newPosition) throw ( lang::exceptions::IllegalArgumentException )  
*Sets this buffer's position.*
- virtual int **limit** () const
- virtual **Buffer** & **limit** (int newLimit) throw ( lang::exceptions::IllegalArgumentException )  
*Sets this buffer's limit.*
- virtual **Buffer** & **mark** ()  
*Sets this buffer's mark at its position.*
- virtual **Buffer** & **reset** () throw ( InvalidMarkException )  
*Resets this buffer's position to the previously-marked position.*
- virtual **Buffer** & **clear** ()  
*Clears this buffer.*
- virtual **Buffer** & **flip** ()



*Flips this buffer.*

- virtual **Buffer** & **rewind** ()

*Rewinds this buffer.*

- virtual int **remaining** () const

*Returns the number of elements between the current position and the limit.*

- virtual bool **hasRemaining** () const

*Tells whether there are any elements between the current position and the limit.*

- virtual bool **isReadOnly** () const =0

*Tells whether or not this buffer is read-only.*

### Protected Attributes

- int **\_position**
- int **\_capacity**
- int **\_limit**
- int **\_mark**
- bool **\_markSet**

#### 6.158.1 Detailed Description

A container for data of a specific primitive type. A buffer is a linear, finite sequence of elements of a specific primitive type. Aside from its content, the essential properties of a buffer are its capacity, limit, and position:

A buffer's capacity is the number of elements it contains. The capacity of a buffer is never negative and never changes.

A buffer's limit is the index of the first element that should not be read or written. A buffer's limit is never negative and is never greater than its capacity.

A buffer's position is the index of the next element to be read or written. A buffer's position is never negative and is never greater than its limit.

There is one subclass of this class for each non-boolean primitive type.

Transferring data: Each subclass of this class defines two categories of get and put operations: \* Relative operations read or write one or more elements starting at the current position and then increment the position by the number of elements transferred. If the requested transfer exceeds the limit then a relative get operation throws a **BufferUnderflowException** (p. 967) and a relative put operation throws a **BufferOverflowException** (p. 964); in either case, no data is transferred. \* Absolute operations take an explicit element index and do not affect the position. Absolute get and put operations throw an **IndexOutOfBoundsException** if the index argument exceeds the limit.

Data may also, of course, be transferred in to or out of a buffer by the I/O operations of an appropriate channel, which are always relative to the current position.

Marking and resetting:

A buffer's mark is the index to which its position will be reset when the reset method is invoked. The mark is not always defined, but when it is defined it is never negative and is never greater than the position. If the mark is defined then it is discarded when the position or the limit is adjusted to a value smaller than the mark. If the mark is not defined then invoking the reset method causes an **InvalidMarkException** (p. 2204) to be thrown.

Invariants:

The following invariant holds for the mark, position, limit, and capacity values:  $0 \leq \text{mark} \leq \text{position} \leq \text{limit} \leq \text{capacity}$

A newly-created buffer always has a position of zero and a mark that is undefined. The initial limit may be zero, or it may be some other value that depends upon the type of the buffer and the manner in which it is constructed. The initial content of a buffer is, in general, undefined.

Clearing, flipping, and rewinding:

In addition to methods for accessing the position, limit, and capacity values and for marking and resetting, this class also defines the following operations upon buffers:

**clear()** (p. 939) makes a buffer ready for a new sequence of channel-read or relative put operations: It sets the limit to the capacity and the position to zero.

**flip()** (p. 939) makes a buffer ready for a new sequence of channel-write or relative get operations: It sets the limit to the current position and then sets the position to zero.

**rewind()** (p. 942) makes a buffer ready for re-reading the data that it already contains: It leaves the limit unchanged and sets the position to zero.

Read-only buffers:

Every buffer is readable, but not every buffer is writable. The mutation methods of each buffer class are specified as optional operations that will throw a **ReadOnlyBufferException** (p. 3260) when invoked upon a read-only buffer. A read-only buffer does not allow its content to be changed, but its mark, position, and limit values are mutable. Whether or not a buffer is read-only may be determined by invoking its `isReadOnly` method.

Thread safety:

Buffers are not safe for use by multiple concurrent threads. If a buffer is to be used by more than one thread then access to the buffer should be controlled by appropriate synchronization.

Invocation chaining:

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained; for example, the sequence of statements

```
b.flip(); b.position(23); b.limit(42);
```

can be replaced by the single, more compact statement `b.flip().position(23).limit(42);`

## 6.158.2 Constructor & Destructor Documentation

**6.158.2.1** `decaf::nio::Buffer::Buffer ( int capacity )`

**6.158.2.2** `decaf::nio::Buffer::Buffer ( const Buffer & other )`

**6.158.2.3** `virtual decaf::nio::Buffer::~Buffer ( )` [`inline`, `virtual`]

## 6.158.3 Member Function Documentation

**6.158.3.1** `virtual int decaf::nio::Buffer::capacity ( ) const` [`inline`, `virtual`]

### Returns

this buffer's capacity.

**6.158.3.2** `virtual Buffer& decaf::nio::Buffer::clear ( )` [`virtual`]

Clears this buffer.

The position is set to zero, the limit is set to the capacity, and the mark is discarded.

Invoke this method before using a sequence of channel-read or put operations to fill this buffer. For example:

```
buf.clear(); // Prepare buffer for reading in.read(buf); // Read data
```

This method does not actually erase the data in the buffer, but it is named as if it did because it will most often be used in situations in which that might as well be the case.

### Returns

a reference to this buffer.

**6.158.3.3** `virtual Buffer& decaf::nio::Buffer::flip ( )` [`virtual`]

Flips this buffer.

The limit is set to the current position and then the position is set to zero. If the mark is defined then it is discarded.

After a sequence of channel-read or put operations, invoke this method to prepare for a sequence of channel-write or relative get operations. For example:

```
buf.put(magic); // Prepend header in.read(buf); // Read data into rest of buffer buf.flip();  
// Flip buffer out.write(buf); // Write header + data to channel
```

This method is often used in conjunction with the compact method when transferring data from one place to another.

**Returns**

a reference to this buffer.

**6.158.3.4** `virtual bool decaf::nio::Buffer::hasRemaining ( ) const` `[inline, virtual]`

Tells whether there are any elements between the current position and the limit.

**Returns**

true if, and only if, there is at least one element remaining in this buffer.

**6.158.3.5** `virtual bool decaf::nio::Buffer::isReadOnly ( ) const` `[pure virtual]`

Tells whether or not this buffer is read-only.

**Returns**

true if, and only if, this buffer is read-only.

Implemented in `decaf::internal::nio::ByteBuffer` (p. 1030), `decaf::internal::nio::CharArrayBuffer` (p. 1145), `decaf::internal::nio::DoubleArrayBuffer` (p. 1863), `decaf::internal::nio::FloatArrayBuffer` (p. 1983), `decaf::internal::nio::IntArrayBuffer` (p. 2128), `decaf::internal::nio::LongArrayBuffer` (p. 2520), `decaf::internal::nio::ShortArrayBuffer` (p. 3558), and `decaf::nio::ByteBuffer` (p. 1068).

**6.158.3.6** `virtual int decaf::nio::Buffer::limit ( ) const` `[inline, virtual]`

**Returns**

this buffers Limit

**6.158.3.7** `virtual Buffer& decaf::nio::Buffer::limit ( int newLimit ) throw ( lang::exceptions::IllegalArgumentException )` `[virtual]`

Sets this buffer's limit.

If the position is larger than the new limit then it is set to the new limit. If the mark is defined and larger than the new limit then it is discarded.

**Parameters**

<i>newLimit</i>	The new limit value; must be no larger than this buffer's capacity.
-----------------	---

**Returns**

A reference to This buffer

**Exceptions**

<i>IllegalArgumentException</i>	if preconditions on the new pos don't hold.
---------------------------------	---

**6.158.3.8 virtual Buffer& decaf::nio::Buffer::mark ( ) [virtual]**

Sets this buffer's mark at its position.

**Returns**

a reference to this buffer.

**6.158.3.9 virtual int decaf::nio::Buffer::position ( ) const [inline, virtual]****Returns**

the current position in the buffer

**6.158.3.10 virtual Buffer& decaf::nio::Buffer::position ( int *newPosition* ) throw ( lang::exceptions::IllegalArgumentException ) [virtual]**

Sets this buffer's position.

If the mark is defined and larger than the new position then it is discarded.

**Parameters**

<i>newPosition</i>	The new postion in the buffer to set.
--------------------	---------------------------------------

**Returns**

a reference to This buffer.

**Exceptions**

<i>IllegalArgumentException</i>	if preconditions on the new pos don't hold.
---------------------------------	---

**6.158.3.11 virtual int decaf::nio::Buffer::remaining ( ) const [inline, virtual]**

Returns the number of elements between the current position and the limit.

**Returns**

The number of elements remaining in this buffer

**6.158.3.12** `virtual Buffer& decaf::nio::Buffer::reset ( ) throw ( InvalidMarkException )`  
[virtual]

Resets this buffer's position to the previously-marked position.

#### Returns

a reference to this buffer.

#### Exceptions

<i><b>InvalidMarkException</b></i> (p. 2204)	- If the mark has not been set
---	--------------------------------

**6.158.3.13** `virtual Buffer& decaf::nio::Buffer::rewind ( )` [virtual]

Rewinds this buffer.

The position is set to zero and the mark is discarded.

Invoke this method before a sequence of channel-write or get operations, assuming that the limit has already been set appropriately. For example:

```
out.write(buf); // Write remaining data buf.rewind(); // Rewind buffer buf.get(array); //
Copy data into array
```

#### Returns

a reference to this buffer.

### 6.158.4 Field Documentation

**6.158.4.1** `int decaf::nio::Buffer::_capacity` [protected]

**6.158.4.2** `int decaf::nio::Buffer::_limit` [protected]

**6.158.4.3** `int decaf::nio::Buffer::_mark` [protected]

**6.158.4.4** `bool decaf::nio::Buffer::_markSet` [protected]

**6.158.4.5** `int decaf::nio::Buffer::_position` [mutable, protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/Buffer.h`

## 6.159 decaf::io::BufferedInputStream Class Reference

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.

```
#include <src/main/decaf/io/BufferedInputStream.h>
```

Inheritance diagram for decaf::io::BufferedInputStream:

### Public Member Functions

- **BufferedInputStream** (**InputStream** \*stream, bool own=false)

*Constructor.*

- **BufferedInputStream** (**InputStream** \*stream, int bufferSize, bool own=false)  
throw ( lang::exceptions::IllegalArgumentException )

*Constructor.*

- virtual ~**BufferedInputStream** ()
- virtual int **available** () const throw ( decaf::io::IOException )

*Indicates the number of bytes available.*

*The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute. The default implementation of this method returns zero.*

#### Returns

*the number of bytes available on this input stream.*

#### Exceptions

<b>IOException</b> (p. 2210)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

- virtual void **close** () throw ( decaf::io::IOException )

*Closes the **InputStream** (p. 2105) freeing any resources that might have been aquired during the lifetime of this stream.*

*The default implementation of this method does nothing.*

- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Skips over and discards n bytes of data from this input stream.*

*The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.*

*The skip method of **InputStream** (p. 2105) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached.*

*Subclasses are encouraged to provide a more efficient implementation of this method.*

**Parameters**

num	<i>The number of bytes to skip.</i>
-----	-------------------------------------

**Returns**

*total bytes skipped*

**Exceptions**

<b>IOException</b> (p. 2210)	<i>if an I/O error occurs.</i>
<b>UnsupportedOperationException</b>	<i>if the concrete stream class does not support skipping bytes.</i>

- virtual void **mark** (int readLimit)

*Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.*

*If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.*

*Calling mark on a closed stream instance should have no effect.*

*The default implementation of this method does nothing.*

**Parameters**

readLimit	<i>The max bytes read before marked position is invalid.</i>
-----------	--

- virtual void **reset** () throw ( decaf::io::IOException )

*Repositions this stream to the position at the time the mark method was last called on this input stream.*

*If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2210) might be thrown. \* If such an **IOException** (p. 2210) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.*

*If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2210). \* If an **IOException** (p. 2210) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.*

*The default implementation of this method throws an **IOException** (p. 2210).*

**Exceptions**

<b>IOException</b> (p. 2210)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

- virtual bool **markSupported** () const

*Determines if this input stream supports the mark and reset methods.*

*Whether or not mark and reset are supported is an invariant property of a particular input stream instance.*



*The default implementation of this method returns false.*

#### Returns

*true if this stream instance supports marks*

## Protected Member Functions

- virtual int **doReadByte** () throw ( decaf::io::IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

### 6.159.1 Detailed Description

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.

### 6.159.2 Constructor & Destructor Documentation

**6.159.2.1** decaf::io::BufferedInputStream::BufferedInputStream ( InputStream \* *stream*, bool *own* = false )

Constructor.

#### Parameters

<i>stream</i>	The target input stream to buffer.
<i>own</i>	Indicates if we own the stream object, defaults to false.

**6.159.2.2** decaf::io::BufferedInputStream::BufferedInputStream ( InputStream \* *stream*, int *bufferSize*, bool *own* = false ) throw ( lang::exceptions::IllegalArgumentException )

Constructor.

#### Parameters

<i>stream</i>	The target input stream to buffer.
<i>bufferSize</i>	The size in bytes to allocate for the internal buffer.
<i>own</i>	Indicates if we own the stream object, defaults to false.

## Exceptions

<i>IllegalArgumentException</i>	is the size is zero or negative.
---------------------------------	----------------------------------

6.159.2.3 `virtual decaf::io::BufferedInputStream::~BufferedInputStream ( )` [virtual]

### 6.159.3 Member Function Documentation

6.159.3.1 `virtual int decaf::io::BufferedInputStream::available ( ) const throw ( decaf::io::IOException )` [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

#### Returns

the number of bytes available on this input stream.

#### Exceptions

<i><b>IOException</b></i> (p. 2210)	if an I/O error occurs.
--	-------------------------

Reimplemented from `decaf::io::FilterInputStream` (p. 1953).

6.159.3.2 `virtual void decaf::io::BufferedInputStream::close ( ) throw ( decaf::io::IOException )` [virtual]

Closes the **InputStream** (p. 2105) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Reimplemented from `decaf::io::FilterInputStream` (p. 1954).

6.159.3.3 `virtual int decaf::io::BufferedInputStream::doReadArrayBounded ( unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )` [protected, virtual]

Reimplemented from `decaf::io::FilterInputStream` (p. 1954).

6.159.3.4 `virtual int decaf::io::BufferedInputStream::doReadByte ( ) throw ( decaf::io::IOException )` [protected, virtual]

Reimplemented from `decaf::io::FilterInputStream` (p. 1954).

**6.159.3.5** `virtual void decaf::io::BufferedInputStream::mark ( int readLimit )` [virtual]

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

**Parameters**

<i>readLimit</i>	The max bytes read before marked position is invalid.
------------------	---

Reimplemented from **decaf::io::FilterInputStream** (p. 1955).

**6.159.3.6** `virtual bool decaf::io::BufferedInputStream::markSupported ( ) const` [inline, virtual]

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

**Returns**

true if this stream instance supports marks

Reimplemented from **decaf::io::FilterInputStream** (p. 1955).

**6.159.3.7** `virtual void decaf::io::BufferedInputStream::reset ( ) throw ( decaf::io::IOException )` [virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2210) might be thrown. \* If such an **IOException** (p. 2210) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2210). \* If an **IOException** (p. 2210) is not thrown, then the stream

is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2210).

### Exceptions

<b>IOException</b> (p. 2210)	if an I/O error occurs.
---------------------------------	-------------------------

Reimplemented from **decaf::io::FilterInputStream** (p. 1956).

```
6.159.3.8 virtual long long decaf::io::BufferedInputStream::skip
( long long num ) throw ( decaf::io::IOException,
decaf::lang::exceptions::UnsupportedOperationException )
[virtual]
```

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2105) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

### Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

### Returns

total bytes skipped

### Exceptions

<b>IOException</b> (p. 2210)	if an I/O error occurs.
<i>UnsupportedOperationException</i>	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::FilterInputStream** (p. 1956).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**BufferedInputStream.h**

## 6.160 decaf::io::BufferedOutputStream Class Reference

Wrapper around another output stream that buffers output before writing to the target output stream.

```
#include <src/main/decaf/io/BufferedOutputStream.h>
```

Inheritance diagram for decaf::io::BufferedOutputStream:

### Public Member Functions

- **BufferedOutputStream** (**OutputStream** \*stream, bool **own**=false)  
*Constructor.*
- **BufferedOutputStream** (**OutputStream** \*stream, int bufferSize, bool **own**=false)  
throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Constructor.*
- virtual ~**BufferedOutputStream** ()
- virtual void **flush** () throw ( decaf::io::IOException )  
*inheritDoc*
- virtual void **doWriteByte** (unsigned char c) throw ( decaf::io::IOException )
- virtual void **doWriteArray** (const unsigned char \*buffer, int size) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### 6.160.1 Detailed Description

Wrapper around another output stream that buffers output before writing to the target output stream.

### 6.160.2 Constructor & Destructor Documentation

- 6.160.2.1 decaf::io::BufferedOutputStream::BufferedOutputStream ( **OutputStream** \* *stream*, bool *own* = false )

Constructor.

**Parameters**

<i>stream</i>	The target output stream.
<i>own</i>	Indicates if this class owns the stream pointer.

**6.160.2.2** `decaf::io::BufferedOutputStream::BufferedOutputStream ( OutputStream  
* stream, int bufferSize, bool own = false ) throw (`  
`decaf::lang::exceptions::IllegalArgumentException )`

Constructor.

**Parameters**

<i>stream</i>	The target output stream.
<i>bufferSize</i>	The size for the internal buffer.
<i>own</i>	Indicates if this class owns the stream pointer.

**Exceptions**

<i>IllegalArgumentException</i>	if the <i>bufferSize</i> given is negative.
---------------------------------	---

**6.160.2.3** `virtual decaf::io::BufferedOutputStream::~~BufferedOutputStream ( )`  
`[virtual]`

**6.160.3 Member Function Documentation**

**6.160.3.1** `virtual void decaf::io::BufferedOutputStream::doWriteArray ( const unsigned char  
* buffer, int size ) throw ( decaf::io::IOException )` `[protected,`  
`virtual]`

Reimplemented from `decaf::io::FilterOutputStream` (p. 1960).

**6.160.3.2** `virtual void decaf::io::BufferedOutputStream::doWriteArrayBounded (`  
`const unsigned char * buffer, int size, int offset, int length ) throw (`  
`decaf::io::IOException, decaf::lang::exceptions::NullPointerException,`  
`decaf::lang::exceptions::IndexOutOfBoundsException )`  
`[protected, virtual]`

Reimplemented from `decaf::io::FilterOutputStream` (p. 1960).

**6.160.3.3** `virtual void decaf::io::BufferedOutputStream::doWriteByte ( unsigned char c ) throw (`  
`decaf::io::IOException )` `[protected, virtual]`

Reimplemented from `decaf::io::FilterOutputStream` (p. 1960).

6.160.3.4 virtual void decaf::io::BufferedOutputStream::flush ( ) throw ( decaf::io::IOException ) [virtual]

inheritDoc}

Reimplemented from **decaf::io::FilterOutputStream** (p. 1960).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**BufferedOutputStream.h**

## 6.161 decaf::internal::nio::BufferFactory Class Reference

Factory class used by static methods in the **decaf::nio** (p. 147) package to create the various default version of the NIO interfaces.

```
#include <src/main/decaf/internal/nio/BufferFactory.h>
```

### Public Member Functions

- virtual ~**BufferFactory** ()

### Static Public Member Functions

- static **decaf::nio::ByteBuffer** \* **createByteBuffer** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.*

- static **decaf::nio::ByteBuffer** \* **createByteBuffer** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new ByteBuffer.*

- static **decaf::nio::ByteBuffer** \* **createByteBuffer** (std::vector< unsigned char > &buffer)

*Wraps the passed STL Byte Vector in a ByteBuffer.*

- static **decaf::nio::CharBuffer** \* **createCharBuffer** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Allocates a new char buffer whose position will be zero its limit will be its capacity and its mark is not set.*

- static **decaf::nio::CharBuffer** \* **createCharBuffer** (char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new CharBuffer.*

- static **decaf::nio::CharBuffer \* createCharBuffer** (std::vector< char > &buffer)

*Wraps the passed STL Byte Vector in a CharBuffer.*

- static **decaf::nio::DoubleBuffer \* createDoubleBuffer** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Allocates a new double buffer whose position will be zero its limit will be its capacity and its mark is not set.*

- static **decaf::nio::DoubleBuffer \* createDoubleBuffer** (double \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new DoubleBuffer.*

- static **decaf::nio::DoubleBuffer \* createDoubleBuffer** (std::vector< double > &buffer)

*Wraps the passed STL Double Vector in a DoubleBuffer.*

- static **decaf::nio::FloatBuffer \* createFloatBuffer** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Allocates a new float buffer whose position will be zero its limit will be its capacity and its mark is not set.*

- static **decaf::nio::FloatBuffer \* createFloatBuffer** (float \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new FloatBuffer.*

- static **decaf::nio::FloatBuffer \* createFloatBuffer** (std::vector< float > &buffer)

*Wraps the passed STL Float Vector in a FloatBuffer.*

- static **decaf::nio::LongBuffer \* createLongBuffer** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Allocates a new long long buffer whose position will be zero its limit will be its capacity and its mark is not set.*

- static **decaf::nio::LongBuffer \* createLongBuffer** (long long \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new LongBuffer.*

- static **decaf::nio::LongBuffer \* createLongBuffer** (std::vector< long long > &buffer)

*Wraps the passed STL Long Vector in a LongBuffer.*



- static **decaf::nio::IntBuffer \* createIntBuffer** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Allocates a new int buffer whose position will be zero its limit will be its capacity and its mark is not set.*

- static **decaf::nio::IntBuffer \* createIntBuffer** (int \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new IntBuffer.*

- static **decaf::nio::IntBuffer \* createIntBuffer** (std::vector< int > &buffer)

*Wraps the passed STL int Vector in a IntBuffer.*

- static **decaf::nio::ShortBuffer \* createShortBuffer** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Allocates a new short buffer whose position will be zero its limit will be its capacity and its mark is not set.*

- static **decaf::nio::ShortBuffer \* createShortBuffer** (short \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new ShortBuffer.*

- static **decaf::nio::ShortBuffer \* createShortBuffer** (std::vector< short > &buffer)

*Wraps the passed STL Short Vector in a ShortBuffer.*

### 6.161.1 Detailed Description

Factory class used by static methods in the **decaf::nio** (p. 147) package to create the various default version of the NIO interfaces.

**Since**

1.0

## 6.161.2 Constructor & Destructor Documentation

6.161.2.1 `virtual decaf::internal::nio::BufferFactory::~BufferFactory ( ) [inline, virtual]`

## 6.161.3 Member Function Documentation

6.161.3.1 `static decaf::nio::ByteBuffer* decaf::internal::nio::BufferFactory::createByteBuffer ( int capacity ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [static]`

Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.

### Parameters

<i>capacity</i>	The internal buffer's capacity.
-----------------	---------------------------------

### Returns

a newly allocated ByteBuffer which the caller owns.

### Exceptions

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--

6.161.3.2 `static decaf::nio::ByteBuffer* decaf::internal::nio::BufferFactory::createByteBuffer ( unsigned char * buffer, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [static]`

Wraps the passed buffer with a new ByteBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

### Parameters

<i>buffer</i>	The array that will back the new buffer.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

**Returns**

a new ByteBuffer that is backed by buffer, caller owns the returned pointer.

**Exceptions**

<i>NullPointerException</i>	if the buffer given is Null.
<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.

**6.161.3.3** static decaf::nio::ByteBuffer\* decaf::internal::nio::BufferFactory::createByteBuffer ( std::vector< unsigned char > & buffer ) [static]

Wraps the passed STL Byte Vector in a ByteBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters**

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).
---------------	--

**Returns**

a new ByteBuffer that is backed by buffer, caller owns.

**6.161.3.4** static decaf::nio::CharBuffer\* decaf::internal::nio::BufferFactory::createCharBuffer ( int capacity ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [static]

Allocates a new char buffer whose position will be zero its limit will be its capacity and its mark is not set.

**Parameters**

<i>capacity</i>	The internal buffer's capacity.
-----------------	---------------------------------

**Returns**

a newly allocated CharBuffer which the caller owns.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--

**6.161.3.5** `static decaf::nio::CharBuffer* decaf::internal::nio::BufferFactory::createCharBuffer ( char * buffer, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [static]`

Wraps the passed buffer with a new CharBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The array that will back the new buffer.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

#### Returns

a new CharBuffer that is backed by `buffer`, caller owns the returned pointer.

#### Exceptions

<i>NullPointerException</i>	if the buffer given in Null.
<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.

**6.161.3.6** `static decaf::nio::CharBuffer* decaf::internal::nio::BufferFactory::createCharBuffer ( std::vector< char > & buffer ) [static]`

Wraps the passed STL Byte Vector in a CharBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling <code>vector.resize( N )</code> .
---------------	--

#### Returns

a new CharBuffer that is backed by `buffer`, caller owns.

**6.161.3.7** `static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer ( double * buffer, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [static]

Wraps the passed buffer with a new DoubleBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The array that will back the new buffer.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

#### Returns

a new DoubleBuffer that is backed by buffer, caller owns the returned pointer.

#### Exceptions

<i>NullPointerException</i>	if the buffer given in Null.
<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.

**6.161.3.8** `static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer ( std::vector< double > & buffer )` [static]

Wraps the passed STL Double Vector in a DoubleBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).
---------------	--

#### Returns

a new DoubleBuffer that is backed by buffer, caller owns.

**6.161.3.9** `static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer ( int capacity ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[static]

Allocates a new double buffer whose position will be zero its limit will be its capacity and its mark is not set.

#### Parameters

<i>capacity</i>	The internal buffer's capacity.
-----------------	---------------------------------

#### Returns

a newly allocated DoubleBuffer which the caller owns.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--

**6.161.3.10** `static decaf::nio::FloatBuffer* decaf::internal::nio::BufferFactory::createFloatBuffer ( int capacity ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[static]

Allocates a new float buffer whose position will be zero its limit will be its capacity and its mark is not set.

#### Parameters

<i>capacity</i>	The internal buffer's capacity.
-----------------	---------------------------------

#### Returns

a newly allocated FloatBuffer which the caller owns.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--

**6.161.3.11** `static decaf::nio::FloatBuffer* decaf::internal::nio::BufferFactory::createFloatBuffer ( float * buffer, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [static]

Wraps the passed buffer with a new FloatBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The array that will back the new buffer.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

#### Returns

a new FloatBuffer that is backed by `buffer`, caller owns the returned pointer.

#### Exceptions

<i>NullPointerException</i>	if the buffer given is Null.
<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.

**6.161.3.12 static decaf::nio::FloatBuffer\* decaf::internal::nio::BufferFactory::createFloatBuffer (std::vector< float > & *buffer* ) [static]**

Wraps the passed STL Float Vector in a FloatBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling <code>vector.resize( N )</code> .
---------------	--

#### Returns

a new FloatBuffer that is backed by `buffer`, caller owns.

**6.161.3.13** `static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::createIntBuffer ( int capacity )  
throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[static]`

Allocates a new int buffer whose position will be zero its limit will be its capacity and its mark is not set.

#### Parameters

<i>capacity</i>	The internal buffer's capacity.
-----------------	---------------------------------

#### Returns

a newly allocated IntBuffer which the caller owns.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--

**6.161.3.14** `static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::createIntBuffer ( std::vector< int > & buffer ) [static]`

Wraps the passed STL int Vector in a IntBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling <code>vector.resize( N )</code> .
---------------	--

#### Returns

a new IntBuffer that is backed by `buffer`, caller owns.

**6.161.3.15** `static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::createIntBuffer ( int * buffer, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [static]`

Wraps the passed buffer with a new IntBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its



mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The array that will back the new buffer.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

#### Returns

a new IntBuffer that is backed by buffer, caller owns the returned pointer.

#### Exceptions

<i>NullPointerException</i>	if the buffer given in Null.
<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.

**6.161.3.16** `static decaf::nio::LongBuffer* decaf::internal::nio::BufferFactory::createLongBuffer ( std::vector< long long > & buffer ) [static]`

Wraps the passed STL Long Vector in a LongBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).
---------------	--

#### Returns

a new LongBuffer that is backed by buffer, caller owns.

**6.161.3.17** `static decaf::nio::LongBuffer* decaf::internal::nio::BufferFactory::createLongBuffer ( int capacity ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [static]`

Allocates a new long long buffer whose position will be zero its limit will be its capacity and its mark is not set.

**Parameters**

<i>capacity</i>	- the internal buffer's capacity.
-----------------	-----------------------------------

**Returns**

a newly allocated DoubleBuffer which the caller owns.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--

**6.161.3.18** `static decaf::nio::LongBuffer* decaf::internal::nio::BufferFactory::createLongBuffer ( long long * buffer, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [static]

Wraps the passed buffer with a new LongBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters**

<i>buffer</i>	The array that will back the new buffer.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

**Returns**

a new LongBuffer that is backed by buffer, caller owns the returned pointer.

**Exceptions**

<i>NullPointerException</i>	if the buffer given in Null.
<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.

**6.161.3.19** static decaf::nio::ShortBuffer\* decaf::internal::nio::BufferFactory::createShortBuffer ( int *capacity* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [static]

Allocates a new short buffer whose position will be zero its limit will be its capacity and its mark is not set.

#### Parameters

<i>capacity</i>	The internal buffer's capacity.
-----------------	---------------------------------

#### Returns

a newly allocated ShortBuffer which the caller owns.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--

**6.161.3.20** static decaf::nio::ShortBuffer\* decaf::internal::nio::BufferFactory::createShortBuffer ( short \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [static]

Wraps the passed buffer with a new ShortBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The array that will back the new buffer.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

#### Returns

a new ShortBuffer that is backed by buffer, caller owns the returned pointer.

#### Exceptions

<i>NullPointerException</i>	if the buffer given in Null.
<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.

**6.161.3.21** `static decaf::nio::ShortBuffer* decaf::internal::nio::BufferFactory::createShortBuffer (std::vector< short > & buffer ) [static]`

Wraps the passed STL Short Vector in a ShortBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling <code>vector.resize( N )</code> .
---------------	--

#### Returns

a new `DoubleBuffer` that is backed by `buffer`, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/BufferFactory.h`

## 6.162 decaf::nio::BufferOverflowException Class Reference

```
#include <src/main/decaf/nio/BufferOverflowException.h>
```

Inheritance diagram for `decaf::nio::BufferOverflowException`:

#### Public Member Functions

- **BufferOverflowException** () throw ()  
*Default Constructor.*
- **BufferOverflowException** (const `lang::Exception` &ex) throw ()  
*Copy Constructor.*
- **BufferOverflowException** (const **BufferOverflowException** &ex) throw ()  
*Copy Constructor.*
- **BufferOverflowException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **BufferOverflowException** (const std::exception \*cause) throw ()

*Constructor.*

- **BufferOverflowException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()

*Constructor.*

- virtual **BufferOverflowException** \* clone () const

*Clones this exception.*

- virtual ~**BufferOverflowException** () throw ()

### 6.162.1 Constructor & Destructor Documentation

6.162.1.1 decaf::nio::BufferOverflowException::BufferOverflowException ( ) throw ()  
[inline]

Default Constructor.

6.162.1.2 decaf::nio::BufferOverflowException::BufferOverflowException ( const lang::Exception & ex ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

6.162.1.3 decaf::nio::BufferOverflowException::BufferOverflowException ( const BufferOverflowException & ex ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

6.162.1.4 decaf::nio::BufferOverflowException::BufferOverflowException ( const char \* file, const int lineNumber, const std::exception \* cause, const char \* msg, ... ) throw ()  
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.162.1.5** `decaf::nio::BufferOverflowException::BufferOverflowException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.162.1.6** `decaf::nio::BufferOverflowException::BufferOverflowException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor.

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.162.1.7** `virtual decaf::nio::BufferOverflowException::~~BufferOverflowException ( ) throw () [inline, virtual]`

## 6.162.2 Member Function Documentation

**6.162.2.1** `virtual BufferOverflowException* decaf::nio::BufferOverflowException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **decaf::lang::Exception** (p. 1889).

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/BufferOverflowException.h`

## 6.163 decaf::nio::BufferUnderflowException Class Reference

```
#include <src/main/decaf/nio/BufferUnderflowException.h>
```

Inheritance diagram for decaf::nio::BufferUnderflowException:

### Public Member Functions

- **BufferUnderflowException** () throw ()

*Default Constructor.*

- **BufferUnderflowException** (const lang::Exception &ex) throw ()

*Copy Constructor.*

- **BufferUnderflowException** (const **BufferUnderflowException** &ex) throw ()

*Copy Constructor.*

- **BufferUnderflowException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()

*Constructor - Initializes the file name and line number where this message occurred.*

- **BufferUnderflowException** (const std::exception \*cause) throw ()

*Constructor.*

- **BufferUnderflowException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()

*Constructor.*

- virtual **BufferUnderflowException** \* clone () const

*Clones this exception.*

- virtual ~**BufferUnderflowException** () throw ()

### 6.163.1 Constructor & Destructor Documentation

6.163.1.1 decaf::nio::BufferUnderflowException::BufferUnderflowException ( ) throw ()  
[inline]

Default Constructor.

**6.163.1.2** `decaf::nio::BufferUnderflowException::BufferUnderflowException ( const lang::Exception & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

**6.163.1.3** `decaf::nio::BufferUnderflowException::BufferUnderflowException ( const BufferUnderflowException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

**6.163.1.4** `decaf::nio::BufferUnderflowException::BufferUnderflowException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.163.1.5** `decaf::nio::BufferUnderflowException::BufferUnderflowException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--



6.163.1.6 `decaf::nio::BufferUnderflowException::BufferUnderflowException ( const char * file,  
const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor.

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.163.1.7 `virtual decaf::nio::BufferUnderflowException::~~BufferUnderflowException ( ) throw  
() [inline, virtual]`

#### 6.163.2 Member Function Documentation

6.163.2.1 `virtual BufferUnderflowException* decaf::nio::BufferUnderflowException::clone  
( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **decaf::lang::Exception** (p. 1889).

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/BufferUnderflowException.h`

## 6.164 decaf::lang::Byte Class Reference

```
#include <src/main/decaf/lang/Byte.h>
```

Inheritance diagram for decaf::lang::Byte:

#### Public Member Functions

- **Byte** (unsigned char value)
- **Byte** (const std::string &value) throw ( exceptions::NumberFormatException )
- virtual **~Byte** ()
- virtual int **compareTo** (const **Byte** &c) const  
*Compares this **Byte** (p. 969) instance with another.*

- virtual bool **operator==** (const **Byte** &c) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Byte** &c) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual int **compareTo** (const unsigned char &c) const  
*Compares this **Byte** (p. 969) instance with a char type.*
- virtual bool **operator==** (const unsigned char &c) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const unsigned char &c) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- bool **equals** (const **Byte** &c) const
- bool **equals** (const unsigned char &c) const
- std::string **toString** () const
- virtual double **doubleValue** () const  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const  
*Answers the float value which the receiver represents.*
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual short **shortValue** () const  
*Answers the short value which the receiver represents.*
- virtual int **intValue** () const  
*Answers the int value which the receiver represents.*
- virtual long long **longValue** () const  
*Answers the long value which the receiver represents.*

### Static Public Member Functions

- static std::string **toString** (unsigned char value)
- static **Byte decode** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Decodes a **String** (p. 3775) into a **Byte** (p. 969).*

- static unsigned char **parseByte** (const std::string &s, int radix) throw ( exceptions::NumberFormatException )  
*Parses the string argument as a signed unsigned char in the radix specified by the second argument.*
- static unsigned char **parseByte** (const std::string &s) throw ( exceptions::NumberFormatException )  
*Parses the string argument as a signed decimal unsigned char.*
- static **Byte valueOf** (unsigned char value)  
*Returns a **Character** (p. 1126) instance representing the specified char value.*
- static **Byte valueOf** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Returns a **Byte** (p. 969) object holding the value given by the specified std::string.*
- static **Byte valueOf** (const std::string &value, int radix) throw ( exceptions::NumberFormatException )  
*Returns a **Byte** (p. 969) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

### Static Public Attributes

- static const unsigned char **MIN\_VALUE** = 0x7F  
*The minimum value that a unsigned char can take on.*
- static const unsigned char **MAX\_VALUE** = 0x80  
*The maximum value that a unsigned char can take on.*
- static const int **SIZE** = 8  
*The size of the primitive charactor in bits.*

## 6.164.1 Constructor & Destructor Documentation

### 6.164.1.1 decaf::lang::Byte::Byte ( unsigned char value )

#### Parameters

<i>value</i>	- the primitive value to wrap
--------------	-------------------------------

**6.164.1.2** `decaf::lang::Byte::Byte ( const std::string & value ) throw ( exceptions::NumberFormatException )`

#### Parameters

<i>value</i>	- the string to convert to an unsigned char
--------------	---

#### Exceptions

<i>NumberFormatException</i>	
------------------------------	--

**6.164.1.3** `virtual decaf::lang::Byte::~Byte ( ) [inline, virtual]`

### 6.164.2 Member Function Documentation

**6.164.2.1** `virtual unsigned char decaf::lang::Byte::byteValue ( ) const [inline, virtual]`

Answers the byte value which the receiver represents.

#### Returns

byte the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2919).

**6.164.2.2** `virtual int decaf::lang::Byte::compareTo ( const unsigned char & c ) const [inline, virtual]`

Compares this **Byte** (p. 969) instance with a char type.

#### Parameters

<i>c</i>	- the char instance to be compared
----------	------------------------------------

#### Returns

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable< unsigned char >** (p. 1249).

**6.164.2.3** `virtual int decaf::lang::Byte::compareTo ( const Byte & c ) const [inline, virtual]`

Compares this **Byte** (p. 969) instance with another.

**Parameters**

<i>c</i>	- the <b>Byte</b> (p. 969) instance to be compared
----------	--

**Returns**

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **Byte** > (p. 1249).

**6.164.2.4 static Byte decaf::lang::Byte::decode ( const std::string & *value* ) throw ( exceptions::NumberFormatException ) [static]**

Decodes a **String** (p. 3775) into a **Byte** (p. 969).

Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the **Byte::parseByte** (p. 976) method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a **NumberFormatException** will be thrown. The result is negated if first character of the specified **String** (p. 3775) is the minus sign. No whitespace characters are permitted in the string.

**Parameters**

<i>value</i>	- The string to decode
--------------	------------------------

**Returns**

a **Byte** (p. 969) object containing the decoded value

**Exceptions**

<i>NumberFormatException</i>	if the string is not formatted correctly.
------------------------------	---

**6.164.2.5 virtual double decaf::lang::Byte::doubleValue ( ) const [inline, virtual]**

Answers the double value which the receiver represents.

**Returns**

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2919).

**6.164.2.6** `bool decaf::lang::Byte::equals ( const unsigned char & c ) const` `[inline, virtual]`

#### Returns

true if the two Bytes have the same value.

Implements **decaf::lang::Comparable**< unsigned char > (p. 1250).

**6.164.2.7** `bool decaf::lang::Byte::equals ( const Byte & c ) const` `[inline, virtual]`

#### Returns

true if the two **Byte** (p. 969) Objects have the same value.

Implements **decaf::lang::Comparable**< Byte > (p. 1250).

**6.164.2.8** `virtual float decaf::lang::Byte::floatValue ( ) const` `[inline, virtual]`

Answers the float value which the receiver represents.

#### Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2920).

**6.164.2.9** `virtual int decaf::lang::Byte::intValue ( ) const` `[inline, virtual]`

Answers the int value which the receiver represents.

#### Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2920).

**6.164.2.10** `virtual long long decaf::lang::Byte::longValue ( ) const` `[inline, virtual]`

Answers the long value which the receiver represents.

#### Returns

long long the value of the receiver.

Implements **decaf::lang::Number** (p. 2920).

**6.164.2.11** `virtual bool decaf::lang::Byte::operator< ( const unsigned char & c ) const`  
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<code>c</code>	- the value to be compared to this one.
----------------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< unsigned char >** (p. 1250).

**6.164.2.12** `virtual bool decaf::lang::Byte::operator< ( const Byte & c ) const` [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<code>c</code>	- the value to be compared to this one.
----------------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Byte >** (p. 1250).

**6.164.2.13** `virtual bool decaf::lang::Byte::operator== ( const Byte & c ) const` [inline, virtual]

Compares equality between this object and the one passed.

#### Parameters

<code>c</code>	- the value to be compared to this one.
----------------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Byte >** (p. 1250).

**6.164.2.14** `virtual bool decaf::lang::Byte::operator== ( const unsigned char & c ) const`  
`[inline, virtual]`

Compares equality between this object and the one passed.

#### Parameters

<code>c</code>	- the value to be compared to this one.
----------------	---

#### Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< unsigned char >` (p. 1250).

**6.164.2.15** `static unsigned char decaf::lang::Byte::parseByte ( const std::string & s ) throw (`  
`exceptions::NumberFormatException ) [static]`

Parses the string argument as a signed decimal unsigned char.

The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting unsigned char value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseByte( const std::string, int )` method.

#### Parameters

<code>s</code>	- <b>String</b> (p. 3775) to convert to a unsigned char
----------------	---

#### Returns

the converted unsigned char value

#### Exceptions

<i>NumberFormatException</i>	if the string is not a unsigned char.
------------------------------	---------------------------------------

**6.164.2.16** `static unsigned char decaf::lang::Byte::parseByte ( const std::string & s, int radix )`  
`throw ( exceptions::NumberFormatException ) [static]`

Parses the string argument as a signed unsigned char in the radix specified by the second argument.

The characters in the string must all be digits, of the specified radix (as determined by whether `Character.digit(char, int)` (p. 1130) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type `NumberFormatException` is thrown if any of the following situations occurs: \* The first argument is null or is a string of length zero. \* The radix is ei-



ther smaller than **Character::MIN\_RADIX** (p. 1134) or larger than **Character::MAX\_RADIX** (p. 1134). \* Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. \* The value represented by the string is not a value of type unsigned char.

#### Parameters

<i>s</i>	- the <b>String</b> (p. 3775) containing the unsigned char to be parsed
<i>radix</i>	- the radix to be used while parsing s

#### Returns

the unsigned char represented by the string argument in the specified radix.

#### Exceptions

<i>NumberFormatException</i>	- If <b>String</b> (p. 3775) does not contain a parsable unsigned char.
------------------------------	---

**6.164.2.17** virtual short decaf::lang::Byte::shortValue ( ) const [inline, virtual]

Answers the short value which the receiver represents.

#### Returns

short the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2920).

**6.164.2.18** static std::string decaf::lang::Byte::toString ( unsigned char value ) [static]

#### Returns

a string representing the primitive value as Base 10

**6.164.2.19** std::string decaf::lang::Byte::toString ( ) const

#### Returns

this **Byte** (p. 969) Object as a **String** (p. 3775) Representation

**6.164.2.20** static Byte decaf::lang::Byte::valueOf ( unsigned char value ) [inline, static]

Returns a **Character** (p. 1126) instance representing the specified char value.

#### Parameters

<i>value</i>	- the primitive char to wrap.
--------------	-------------------------------

**Returns**

a new **Character** (p. 1126) instance that wraps this value.

**6.164.2.21** `static Byte decaf::lang::Byte::valueOf ( const std::string & value, int radix ) throw ( exceptions::NumberFormatException ) [static]`

Returns a **Byte** (p. 969) object holding the value extracted from the specified `std::string` when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed unsigned char in the radix specified by the second argument, exactly as if the argument were given to the `parseByte( std::string, int )` method. The result is a **Byte** (p. 969) object that represents the unsigned char value specified by the string.

**Parameters**

<i>value</i>	- <code>std::string</code> to parse as base ( <i>radix</i> )
<i>radix</i>	- base of the string to parse.

**Returns**

new **Byte** (p. 969) Object wrapping the primitive

**Exceptions**

<i>NumberFormatException</i>	if the string is not a valid unsigned char.
------------------------------	---

**6.164.2.22** `static Byte decaf::lang::Byte::valueOf ( const std::string & value ) throw ( exceptions::NumberFormatException ) [static]`

Returns a **Byte** (p. 969) object holding the value given by the specified `std::string`.

The argument is interpreted as representing a signed decimal unsigned char, exactly as if the argument were given to the `parseByte( std::string )` method. The result is a **Byte** (p. 969) object that represents the unsigned char value specified by the string.

**Parameters**

<i>value</i>	- <code>std::string</code> to parse as base 10
--------------	--

**Returns**

new **Byte** (p. 969) Object wrapping the primitive

**Exceptions**

<i>NumberFormatException</i>	if the string is not a decimal unsigned char.
------------------------------	---

**6.164.3 Field Documentation**

**6.164.3.1** `const unsigned char decaf::lang::Byte::MAX_VALUE = 0x80` [static]

The maximum value that a unsigned char can take on.

**6.164.3.2** `const unsigned char decaf::lang::Byte::MIN_VALUE = 0x7F` [static]

The minimum value that a unsigned char can take on.

**6.164.3.3** `const int decaf::lang::Byte::SIZE = 8` [static]

The size of the primitive charactor in bits.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Byte.h`

**6.165 decaf::internal::util::ByteArrayAdapter Class Reference**

This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.

```
#include <src/main/decaf/internal/util/ByteArrayAdapter.h>
```

**Data Structures**

- union **Array**

**Public Member Functions**

- **ByteArrayAdapter** (int size) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted.*
- **ByteArrayAdapter** (unsigned char \*array, int size, bool own=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a byte array object that wraps the given array.*

- **ByteArrayAdapter** (char \*array, int size, bool own=false) throw ( decaf::lang::exceptions::NullPointerException  
decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a byte array object that wraps the given array.*
- **ByteArrayAdapter** (double \*array, int size, bool own=false) throw ( decaf::lang::exceptions::NullPointerException  
decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a byte array object that wraps the given array.*
- **ByteArrayAdapter** (float \*array, int size, bool own=false) throw ( decaf::lang::exceptions::NullPointerException  
decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a byte array object that wraps the given array.*
- **ByteArrayAdapter** (long long \*array, int size, bool own=false) throw ( de-  
caf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException  
)  
*Creates a byte array object that wraps the given array.*
- **ByteArrayAdapter** (int \*array, int size, bool own=false) throw ( decaf::lang::exceptions::NullPointerException  
decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a byte array object that wraps the given array.*
- **ByteArrayAdapter** (short \*array, int size, bool own=false) throw ( decaf::lang::exceptions::NullPointerException  
decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a byte array object that wraps the given array.*
- virtual ~**ByteArrayAdapter** ()
- virtual int **getCapacity** () const  
*Gets the size of the underlying array.*
- virtual int **getCharCapacity** () const  
*Gets the size of the underlying array as if it contains chars.*
- virtual int **getDoubleCapacity** () const  
*Gets the size of the underlying array as if it contains doubles.*
- virtual int **getFloatCapacity** () const  
*Gets the size of the underlying array as if it contains doubles.*
- virtual int **getLongCapacity** () const  
*Gets the size of the underlying array as if it contains doubles.*
- virtual int **getIntCapacity** () const  
*Gets the size of the underlying array as if it contains ints.*
- virtual int **getShortCapacity** () const  
*Gets the size of the underlying array as if it contains shorts.*

- virtual unsigned char \* **getByteArray** ()  
*Gets the pointer to the array we are wrapping.*
- virtual char \* **getCharArray** ()  
*Gets the pointer to the array we are wrapping.*
- virtual short \* **getShortArray** ()  
*Gets the pointer to the array we are wrapping.*
- virtual int \* **getIntArray** ()  
*Gets the pointer to the array we are wrapping.*
- virtual long long \* **getLongArray** ()  
*Gets the pointer to the array we are wrapping.*
- virtual double \* **getDoubleArray** ()  
*Gets the pointer to the array we are wrapping.*
- virtual float \* **getFloatArray** ()  
*Gets the pointer to the array we are wrapping.*
- virtual void **read** (unsigned char \*buffer, int size, int offset, int length) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException, decaf::nio::BufferUnderflowException )  
*Reads from the Byte array starting at the specified offset and reading the specified length.*
- virtual void **write** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException, decaf::nio::BufferOverflowException )  
*Writes from the Byte array given, starting at the specified offset and writing the specified amount of data into this objects internal array.*
- virtual void **resize** (int size) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::InvalidStateException )  
*Resizes the underlying array to the new given size, preserving all the Data that was previously in the array, unless the resize is smaller than the current size in which case only the data that will fit into the new array is preserved.*
- virtual void **clear** () throw ()  
*Clear all data from that Array, setting the underlying bytes to zero.*
- unsigned char & **operator** [] (int index) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Allows the **ByteArrayAdapter** (p. 979) to be indexed as a standard array.*

- const unsigned char & **operator[]** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual unsigned char **get** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Absolute get method.*
- virtual char **getChar** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads one byte at the given index and returns it.*
- virtual double **getDouble** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads eight bytes at the given index and returns it.*
- virtual double **getDoubleAt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads eight bytes at the given byte index and returns it.*
- virtual float **getFloat** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads four bytes at the given index and returns it.*
- virtual float **getFloatAt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads four bytes at the given byte index and returns it.*
- virtual long long **getLong** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads eight bytes at the given index and returns it.*
- virtual long long **getLongAt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads eight bytes at the given byte index and returns it.*
- virtual int **getInt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads four bytes at the given index and returns it.*
- virtual int **getIntAt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads four bytes at the given byte index and returns it.*
- virtual short **getShort** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads two bytes at the given index and returns it.*

- virtual short **getShortAt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads two bytes at the given byte index and returns it.*
- virtual **ByteArrayAdapter** & **put** (int index, unsigned char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes the given byte into this buffer at the given index.*
- virtual **ByteArrayAdapter** & **putChar** (int index, char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes one byte containing the given value, into this buffer at the given index.*
- virtual **ByteArrayAdapter** & **putDouble** (int index, double value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes eight bytes containing the given value, into this buffer at the given index.*
- virtual **ByteArrayAdapter** & **putDoubleAt** (int index, double value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes eight bytes containing the given value, into this buffer at the given byte index.*
- virtual **ByteArrayAdapter** & **putFloat** (int index, float value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes four bytes containing the given value, into this buffer at the given index.*
- virtual **ByteArrayAdapter** & **putFloatAt** (int index, float value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes four bytes containing the given value, into this buffer at the given byte index.*
- virtual **ByteArrayAdapter** & **putLong** (int index, long long value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes eight bytes containing the given value, into this buffer at the given index.*
- virtual **ByteArrayAdapter** & **putLongAt** (int index, long long value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes eight bytes containing the given value, into this buffer at the given byte index.*
- virtual **ByteArrayAdapter** & **putInt** (int index, int value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes four bytes containing the given value, into this buffer at the given index.*
- virtual **ByteArrayAdapter** & **putIntAt** (int index, int value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes four bytes containing the given value, into this buffer at the given byte index.*
- virtual **ByteArrayAdapter** & **putShort** (int index, short value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes two bytes containing the given value, into this buffer at the given index.*

- virtual **ByteArrayAdapter** & **putShortAt** (int index, short value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Writes two bytes containing the given value, into this buffer at the given byte index.*

### 6.165.1 Detailed Description

This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data. All the array types are mapped down to a byte array and methods are supplied for accessing the data in any of the primitive type forms.

Methods in this class that do not return a specific value return a reference to this object so that calls can be chained.

#### Since

1.0

### 6.165.2 Constructor & Destructor Documentation

#### 6.165.2.1 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter ( int *size* ) throw ( decaf::lang::exceptions::IllegalArgumentException )

Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

#### Parameters

<i>size</i>	The size of the array, this is the limit we read and write to.
-------------	--

#### Exceptions

<i>IllegalArgumentException</i>	if size is negative.
---------------------------------	----------------------

#### 6.165.2.2 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter ( unsigned char \* *array*, int *size*, bool *own* = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.



**Parameters**

<i>array</i>	The physical array to wrap.
<i>size</i>	The size of the array, this is the limit we read and write to.
<i>own</i>	Indicates if this class is now the owner of the pointer.

**Exceptions**

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the size is negative.

**6.165.2.3** `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter ( char * array, int size, bool own = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

**Parameters**

<i>array</i>	The physical array to wrap.
<i>size</i>	The size of the array, this is the limit we read and write to.
<i>own</i>	Indicates if this class is now the owner of the pointer.

**Exceptions**

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the size is negative.

**6.165.2.4** `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter ( double * array, int size, bool own = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

**Parameters**

<i>array</i>	The physical array to wrap.
<i>size</i>	The size of the array, this is the limit we read and write to.
<i>own</i>	Indicates if this class is now the owner of the pointer.

**Exceptions**

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the size is negative.

**6.165.2.5** `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter ( float * array, int size, bool own = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

**Parameters**

<i>array</i>	The physical array to wrap.
<i>size</i>	The size of the array, this is the limit we read and write to.
<i>own</i>	Indicates if this class is now the owner of the pointer.

**Exceptions**

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the size is negative.

**6.165.2.6** `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter ( long long * array, int size, bool own = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

**Parameters**

<i>array</i>	The physical array to wrap.
<i>size</i>	The size of the array, this is the limit we read and write to.
<i>own</i>	Indicates if this class is now the owner of the pointer.

**Exceptions**

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the size is negative.

**6.165.2.7** `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter ( int * array, int size, bool own = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

#### Parameters

<i>array</i>	The physical array to wrap.
<i>size</i>	The size of the array, this is the limit we read and write to.
<i>own</i>	Indicates if this class is now the owner of the pointer.

#### Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the size is negative.

**6.165.2.8** `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter ( short * array, int size, bool own = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

#### Parameters

<i>array</i>	The physical array to wrap.
<i>size</i>	The size of the array, this is the limit we read and write to.
<i>own</i>	Indicates if this class is now the owner of the pointer.

#### Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the size is negative.

6.165.2.9 `virtual decaf::internal::util::ByteArrayAdapter::~~ByteArrayAdapter ( )`  
[virtual]

### 6.165.3 Member Function Documentation

6.165.3.1 `virtual void decaf::internal::util::ByteArrayAdapter::clear ( ) throw ()` [virtual]

Clear all data from that Array, setting the underlying bytes to zero.

6.165.3.2 `virtual unsigned char decaf::internal::util::ByteArrayAdapter::get ( int index )`  
`const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Absolute get method.

Reads the byte at the given index.

#### Parameters

<i>index</i>	The index in the Buffer where the byte is to be read.
--------------	---

#### Returns

the byte that is located at the given index.

#### Exceptions

<i>IndexOutOfBoundsException</i>	If index is not smaller than the buffer's limit or is negative.
----------------------------------	---

6.165.3.3 `virtual unsigned char* decaf::internal::util::ByteArrayAdapter::getByteArray ( )`  
[inline, virtual]

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 979) objects that point to this array.

#### Returns

an unsigned char\* pointer to the array this object wraps.

6.165.3.4 `virtual int decaf::internal::util::ByteArrayAdapter::getCapacity ( ) const`  
[inline, virtual]

Gets the size of the underlying array.

#### Returns

the size the array.

**6.165.3.5** `virtual char decaf::internal::util::ByteArrayAdapter::getChar ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Reads one byte at the given index and returns it.

#### Parameters

<i>index</i>	The index in the Buffer where the byte is to be read.
--------------	---

#### Returns

the byte that is located at the given index.

#### Exceptions

<i>IndexOutOfBoundsException</i>	If index is not smaller than the buffer's limit or is negative.
----------------------------------	---

**6.165.3.6** `virtual char* decaf::internal::util::ByteArrayAdapter::getCharArray ( )` [inline, virtual]

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 979) objects that point to this array.

#### Returns

an char\* pointer to the array this object wraps.

**6.165.3.7** `virtual int decaf::internal::util::ByteArrayAdapter::getCharCapacity ( ) const` [inline, virtual]

Gets the size of the underlying array as if it contains chars.

#### Returns

the size the array.

**6.165.3.8** `virtual double decaf::internal::util::ByteArrayAdapter::getDouble ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Reads eight bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters**

<i>index</i>	The index in the Buffer where the bytes are to be read.
--------------	---

**Returns**

the value at the given index in the buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

**6.165.3.9** `virtual double* decaf::internal::util::ByteArrayAdapter::getDoubleArray ( )`  
`[inline, virtual]`

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 979) objects that point to this array.

**Returns**

an double\* pointer to the array this object wraps.

**6.165.3.10** `virtual double decaf::internal::util::ByteArrayAdapter::getDoubleAt ( int index )`  
`const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
`[virtual]`

Reads eight bytes at the given byte index and returns it.

**Parameters**

<i>index</i>	The index in the Buffer where the bytes are to be read
--------------	--

**Returns**

the value at the given index in the buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

**6.165.3.11** `virtual int decaf::internal::util::ByteArrayAdapter::getDoubleCapacity ( ) const`  
`[inline, virtual]`

Gets the size of the underlying array as if it contains doubles.

**Returns**

the size the array.

**6.165.3.12** `virtual float decaf::internal::util::ByteArrayAdapter::getFloat ( int index ) const  
throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]`

Reads four bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters**

<i>index</i>	The index in the Buffer where the bytes are to be read.
--------------	---

**Returns**

the value at the given index in the buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

**6.165.3.13** `virtual float* decaf::internal::util::ByteArrayAdapter::getFloatArray ( )  
[inline, virtual]`

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 979) objects that point to this array.

**Returns**

an float\* pointer to the array this object wraps.

**6.165.3.14** `virtual float decaf::internal::util::ByteArrayAdapter::getFloatAt ( int index )  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]`

Reads four bytes at the given byte index and returns it.

**Parameters**

<i>index</i>	The index in the Buffer where the bytes are to be read
--------------	--

**Returns**

the value at the given index in the buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

**6.165.3.15** `virtual int decaf::internal::util::ByteArrayAdapter::getFloatCapacity ( ) const`  
`[inline, virtual]`

Gets the size of the underlying array as if it contains doubles.

**Returns**

the size the array.

**6.165.3.16** `virtual int decaf::internal::util::ByteArrayAdapter::getInt ( int index ) const throw (`  
`decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Reads four bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters**

<i>index</i>	The index in the Buffer where the bytes are to be read.
--------------	---

**Returns**

the value at the given index in the buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

**6.165.3.17** `virtual int* decaf::internal::util::ByteArrayAdapter::getIntArray ( ) [inline,`  
`virtual]`

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 979) objects that point to this array.



**Returns**

an int\* pointer to the array this object wraps.

**6.165.3.18** `virtual int decaf::internal::util::ByteArrayAdapter::getIntAt ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Reads four bytes at the given byte index and returns it.

**Parameters**

<i>index</i>	The index in the Buffer where the bytes are to be read
--------------	--

**Returns**

the value at the given index in the buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

**6.165.3.19** `virtual int decaf::internal::util::ByteArrayAdapter::getIntCapacity ( ) const` [inline, virtual]

Gets the size of the underlying array as if it contains ints.

**Returns**

the size the array.

**6.165.3.20** `virtual long long decaf::internal::util::ByteArrayAdapter::getLong ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Reads eight bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters**

<i>index</i>	The index in the Buffer where the bytes are to be read.
--------------	---

**Returns**

the value at the given index in the buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

**6.165.3.21** `virtual long long* decaf::internal::util::ByteArrayAdapter::getLongArray ( )`  
`[inline, virtual]`

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 979) objects that point to this array.

**Returns**

an long long\* pointer to the array this object wraps.

**6.165.3.22** `virtual long long decaf::internal::util::ByteArrayAdapter::getLongAt ( int index )`  
`const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
`[virtual]`

Reads eight bytes at the given byte index and returns it.

**Parameters**

<i>index</i>	The index in the Buffer where the bytes are to be read
--------------	--

**Returns**

the value at the given index in the buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

**6.165.3.23** `virtual int decaf::internal::util::ByteArrayAdapter::getLongCapacity ( ) const`  
`[inline, virtual]`

Gets the size of the underlying array as if it contains doubles.

**Returns**

the size the array.

**6.165.3.24** `virtual short decaf::internal::util::ByteArrayAdapter::getShort ( int index ) const  
throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]`

Reads two bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

#### Parameters

<i>index</i>	The index in the Buffer where the bytes are to be read.
--------------	---

#### Returns

the value at the given index in the buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

**6.165.3.25** `virtual short* decaf::internal::util::ByteArrayAdapter::getShortArray ( )  
[inline, virtual]`

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 979) objects that point to this array.

#### Returns

an short\* pointer to the array this object wraps.

**6.165.3.26** `virtual short decaf::internal::util::ByteArrayAdapter::getShortAt ( int index )  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]`

Reads two bytes at the given byte index and returns it.

#### Parameters

<i>index</i>	The index in the Buffer where the bytes are to be read
--------------	--

#### Returns

the value at the given index in the buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

**6.165.3.27** `virtual int decaf::internal::util::ByteArrayAdapter::getShortCapacity ( ) const`  
`[inline, virtual]`

Gets the size of the underlying array as if it contains shorts.

**Returns**

the size the array.

**6.165.3.28** `unsigned char& decaf::internal::util::ByteArrayAdapter::operator[] ( int index ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`

Allows the **ByteArrayAdapter** (p. 979) to be indexed as a standard array.  
calling the non constant version allows the user to change the value at index

**Parameters**

<i>index</i>	The position in the array to access, if the value is negative or greater than the size of the underlying array an <code>IndexOutOfBoundsException</code> is thrown.
--------------	---

**Exceptions**

<i>IndexOutOfBoundsException</i>	if the preconditions of index are not met.
----------------------------------	--

**6.165.3.29** `const unsigned char& decaf::internal::util::ByteArrayAdapter::operator[] ( int index )`  
`const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`

**6.165.3.30** `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::put`  
`( int index, unsigned char value ) throw (`  
`decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Writes the given byte into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters**

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write to the array.

**Returns**

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

**6.165.3.31** `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putChar  
( int index, char value ) throw ( de-  
caf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]`

Writes one byte containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters**

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write to the array.

**Returns**

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

**6.165.3.32** `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putDouble  
( int index, double value ) throw ( de-  
caf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]`

Writes eight bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters**

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write to the array.

**Returns**

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

**6.165.3.33** `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putDoubleAt ( int index, double value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Writes eight bytes containing the given value, into this buffer at the given byte index.

**Parameters**

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write.

**Returns**

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

**6.165.3.34** `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putFloat ( int index, float value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Writes four bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters**

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write to the array.

**Returns**

a reference to this buffer.

### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

**6.165.3.35** virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putFloatAt ( int *index*, float *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Writes four bytes containing the given value, into this buffer at the given byte index.

### Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write.

### Returns

a reference to this buffer.

### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

**6.165.3.36** virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putInt ( int *index*, int *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Writes four bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

### Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write to the array.

### Returns

a reference to this buffer.

### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

**6.165.3.37** `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putIntAt  
( int index, int value ) throw ( de-  
caf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Writes four bytes containing the given value, into this buffer at the given byte index.

#### Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write.

#### Returns

a reference to this buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

**6.165.3.38** `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putLong  
( int index, long long value ) throw ( de-  
caf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

#### Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write to the array.

#### Returns

a reference to this buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--



**6.165.3.39** virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putLongAt  
( int *index*, long long *value* ) throw ( de-  
caf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Writes eight bytes containing the given value, into this buffer at the given byte index.

#### Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write.

#### Returns

a reference to this buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

**6.165.3.40** virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putShort  
( int *index*, short *value* ) throw ( de-  
caf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Writes two bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

#### Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write to the array.

#### Returns

a reference to this buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

**6.165.3.41** `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putShortAt ( int index, short value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Writes two bytes containing the given value, into this buffer at the given byte index.

#### Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write.

#### Returns

a reference to this buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

**6.165.3.42** `virtual void decaf::internal::util::ByteArrayAdapter::read ( unsigned char * buffer, int size, int offset, int length ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException, decaf::nio::BufferUnderflowException )` [virtual]

Reads from the Byte array starting at the specified offset and reading the specified length.

If the length is greater than the size of this underlying byte array then an BufferUnderflowException is thrown.

#### Parameters

<i>buffer</i>	The buffer to read data from this array into.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The position in this array to start reading from.
<i>length</i>	The amount of data to read from this array.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if the offset + length exceeds the size.
<i>NullPointerException</i>	if buffer is null
<i>BufferUnderflowException</i>	if there is not enough data to read because the offset or the length is greater than the size of this array.

**6.165.3.43** virtual void decaf::internal::util::ByteArrayAdapter::resize ( int *size* )  
 throw ( decaf::lang::exceptions::IllegalArgumentException,  
 decaf::lang::exceptions::InvalidStateException ) [virtual]

Resizes the underlying array to the new given size, preserving all the Data that was previously in the array, unless the resize is smaller than the current size in which case only the data that will fit into the new array is preserved.

A **ByteArrayAdapter** (p. 979) can only be resized when it owns the underlying array, if it does not then it will throw an InvalidStateException.

#### Parameters

<i>size</i>	The new size of the array.
-------------	----------------------------

#### Exceptions

<i>IllegalArgumentException</i>	if the size parameter is negative.
<i>InvalidStateException</i>	if this object does not own the buffer.

**6.165.3.44** virtual void decaf::internal::util::ByteArrayAdapter::write ( unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException,  
 decaf::lang::exceptions::NullPointerException,  
 decaf::nio::BufferOverflowException ) [virtual]

Writes from the Byte array given, starting at the specified offset and writing the specified amount of data into this objects internal array.

. If the length is greater than the size of this underlying byte array then an BufferOverflowException is thrown.

#### Parameters

<i>buffer</i>	The buffer to read data from this array into.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The position in this array to start reading from.
<i>length</i>	The amount of data to read from this array.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if the offset + length exceeds the size.
<i>NullPointerException</i>	if buffer is null
<i>BufferOverflowException</i>	if the amount of data to be written to this array or the offset given are larger than this array's size.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/ByteArrayAdapter.h`

## 6.166 decaf::internal::nio::ByteBuffer Class Reference

This class defines six categories of operations upon byte buffers:

```
#include <src/main/decaf/internal/nio/ByteBuffer.h>
```

Inheritance diagram for `decaf::internal::nio::ByteBuffer`:

### Public Member Functions

- **ByteBuffer** (int capacity, bool readOnly=false) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **ByteBuffer** (p. 1004) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **ByteBuffer** (unsigned char \*array, int size, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a **ByteBuffer** (p. 1004) object that wraps the given array.*
- **ByteBuffer** (const decaf::lang::Pointer< ByteArrayAdapter > &array, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.*
- **ByteBuffer** (const ByteBuffer &other)  
*Create a **ByteBuffer** (p. 1004) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*
- virtual ~ByteBuffer ()
- virtual bool isReadOnly () const  
*Tells whether or not this buffer is read-only.*  
**Returns**  
*true if, and only if, this buffer is read-only*
- virtual unsigned char \* array () throw ( decaf::nio::ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException )

*Returns the byte array that backs this buffer.*

*Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.*

*Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.*

#### Returns

*The array that backs this buffer*

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3260)	<i>if this buffer is backed by an array but is read-only</i>
UnsupportedOperationException	<i>if this buffer is not backed by an accessible array</i>

- virtual int **arrayOffset** () const throw ( decaf::nio::ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException )

*Returns the offset within this buffer's backing array of the first element of the buffer.*

*If this buffer is backed by an array then buffer position `p` corresponds to array index `p + arrayOffset()` (p. 1057).*

*Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.*

#### Returns

*The offset within this buffer's array of the first element of the buffer.*

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3260)	<i>if this buffer is backed by an array but is read-only.</i>
UnsupportedOperationException	<i>if this buffer is not backed by an accessible array.</i>

- virtual bool **hasArray** () const

*Tells whether or not this buffer is backed by an accessible byte array.*

*If this method returns `true` then the `array` and `arrayOffset` methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns `true`.*

#### Returns

*`true` if, and only if, this buffer is backed by an array and is not read-only.*

- virtual decaf::nio::CharBuffer \* **asCharBuffer** () const

*Creates a view of this byte buffer as a char buffer.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns**

*the new Char **Buffer** (p. 936), which the caller then owns.*

- virtual **decaf::nio::DoubleBuffer \* asDoubleBuffer () const**

*Creates a view of this byte buffer as a double buffer.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns**

*the new double **Buffer** (p. 936), which the caller then owns.*

- virtual **decaf::nio::FloatBuffer \* asFloatBuffer () const**

*Creates a view of this byte buffer as a float buffer.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns**

*the new float **Buffer** (p. 936), which the caller then owns.*

- virtual **decaf::nio::IntBuffer \* asIntBuffer () const**

*Creates a view of this byte buffer as a int buffer.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns**

*the new int **Buffer** (p. 936), which the caller then owns.*

- virtual **decaf::nio::LongBuffer \* asLongBuffer () const**

*Creates a view of this byte buffer as a long buffer.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns**

*the new long **Buffer** (p. 936), which the caller then owns.*

- virtual **decaf::nio::ShortBuffer \* asShortBuffer () const**

*Creates a view of this byte buffer as a short buffer.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

#### Returns

*the new short **Buffer** (p. 936), which the caller then owns.*

- virtual **ByteBuffer \* asReadOnlyBuffer ()** const

*Creates a new, read-only byte buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.*

*If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.*

*The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.*

#### Returns

*The new, read-only byte buffer which the caller then owns.*

- virtual **ByteBuffer & compact ()** throw ( decaf::nio::ReadOnlyBufferException )

*Compacts this buffer.*

*The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 941) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 940) - 1 is copied to index  $n = \text{limit}()$  (p. 940) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.*

*The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.*

#### Returns

*a reference to this **ByteBuffer** (p. 1049).*

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only.
---	------------------------------

- virtual **ByteBuffer \* duplicate ()**

*Creates a new byte buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns**

a new Byte **Buffer** (p. 936) which the caller owns.

- virtual unsigned char **get** () const throw ( decaf::nio::BufferUnderflowException )

*Relative get method.*

*Reads the byte at this buffer's current position, and then increments the position.*

**Returns**

*The byte at the buffer's current position.*

**Exceptions**

<b>BufferUnderflowException</b> (p. 967)	if the buffer's current position is not smaller than its limit.
---	---

- virtual unsigned char **get** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Absolute get method.*

*Reads the byte at the given index.*

**Parameters**

index	The index in the <b>Buffer</b> (p. 936) where the byte is to be read.
-------	---

**Returns**

*the byte that is located at the given index.*

**Exceptions**

IndexOutOfBoundsException	if index is not smaller than the buffer's limit, or index is negative.
---------------------------	--

- virtual char **getChar** () throw ( decaf::nio::BufferUnderflowException )

*Reads the next byte at this buffer's current position, and then increments the position by one.*

**Returns**

*the next char in the buffer.*

**Exceptions**

<b>BufferUnderflowException</b> (p. 967)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

- virtual char **getChar** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads one byte at the given index and returns it.*

**Parameters**

index	The index in the <b>Buffer</b> (p. 936) where the byte is to be read.
-------	---



**Returns**

*the char at the given index in the buffer*

**Exceptions**

IndexOutOfBoundsException	<i>if index is not smaller than the buffer's limit, or index is negative.</i>
---------------------------	---

- virtual double **getDouble** () throw ( decaf::nio::BufferUnderflowException )

*Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.*

**Returns**

*the next double in the buffer.*

**Exceptions**

<b>BufferUnderflowException</b> (p. 967)	<i>if there are no more bytes remaining in this buffer, meaning we have reached the set limit.</i>
---	--

- virtual double **getDouble** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads eight bytes at the given index and returns it.*

**Parameters**

index	<i>The index in the <b>Buffer</b> (p. 936) where the bytes are to be read.</i>
-------	--

**Returns**

*the double at the given index in the buffer.*

**Exceptions**

IndexOutOfBoundsException	<i>if index is not smaller than the buffer's limit, or index is negative.</i>
---------------------------	---

- virtual float **getFloat** () throw ( decaf::nio::BufferUnderflowException )

*Reads the next four bytes at this buffer's current position, and then increments the position by that amount.*

**Returns**

*the next float in the buffer.*

**Exceptions**

<b>BufferUnderflowException</b> (p. 967)	<i>if there are no more bytes remaining in this buffer, meaning we have reached the set limit.</i>
---	--

- virtual float **getFloat** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads four bytes at the given index and returns it.*

**Parameters**

index	<i>The index in the <b>Buffer</b> (p. 936) where the bytes are to be read.</i>
-------	--

**Returns**

*the float at the given index in the buffer.*

**Exceptions**

IndexOutOfBoundsException	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
---------------------------	--

- virtual long long **getLong** () throw ( decaf::nio::BufferUnderflowException )

*Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.*

**Returns**

*the next long long in the buffer.*

**Exceptions**

<b>BufferUnderflowException</b> (p. 967)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

- virtual long long **getLong** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads eight bytes at the given index and returns it.*

**Parameters**

index	The index in the <b>Buffer</b> (p. 936) where the bytes are to be read.
-------	---

**Returns**

*the long long at the given index in the buffer.*

**Exceptions**

IndexOutOfBoundsException	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
---------------------------	--

- virtual int **getInt** () throw ( decaf::nio::BufferUnderflowException )

*Reads the next four bytes at this buffer's current position, and then increments the position by that amount.*

**Returns**

*the next int in the buffer.*

**Exceptions**

<b>BufferUnderflowException</b> (p. 967)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

- virtual int **getInt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads four bytes at the given index and returns it.*

**Parameters**

index	The index in the <b>Buffer</b> (p. 936) where the bytes are to be read.
-------	---

**Returns**

*the int at the given index in the buffer.*

**Exceptions**

IndexOutOfBoundsException	<i>if there are not enough bytes remaining to fill the requested Data Type, or index is negative.</i>
---------------------------	---

- virtual short **getShort** () throw ( decaf::nio::BufferUnderflowException )

*Reads the next two bytes at this buffer's current position, and then increments the position by that amount.*

**Returns**

*the next short in the buffer.*

**Exceptions**

<b>BufferUnderflowException</b> (p. 967)	<i>if there are no more bytes remaining in this buffer, meaning we have reached the set limit.</i>
---	--

- virtual short **getShort** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads two bytes at the given index and returns it.*

**Parameters**

index	<i>The index in the <b>Buffer</b> (p. 936) where the bytes are to be read.</i>
-------	--

**Returns**

*the short at the given index in the buffer.*

**Exceptions**

IndexOutOfBoundsException	<i>if there are not enough bytes remaining to fill the requested Data Type, or index is negative.</i>
---------------------------	---

- virtual **ByteBuffer** & **put** (unsigned char value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes the given byte into this buffer at the current position, and then increments the position.*

**Parameters**

value	<i>- the byte value to be written.</i>
-------	--

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 964)	<i>if this buffer's current position is not smaller than its limit.</i>
<b>ReadOnlyBufferException</b> (p. 3260)	<i>if this buffer is read-only.</i>

- virtual **ByteBuffer** & **put** (int index, unsigned char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes the given byte into this buffer at the given index.*

**Parameters**

index	- position in the <b>Buffer</b> (p. 936) to write the data
value	- the byte to write.

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>IndexOutOfBoundsException</b>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only.

- virtual **ByteBuffer** & **putChar** (char value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.*

**Parameters**

value	The value to be written.
-------	--------------------------

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 964)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only

- virtual **ByteBuffer** & **putChar** (int index, char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes one byte containing the given value, into this buffer at the given index.*

**Parameters**

index	The position in the <b>Buffer</b> (p. 936) to write the data.
value	The value to write.

**Returns**

*a reference to this buffer.*

**Exceptions**

IndexOutOfBoundsException	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only

- virtual **ByteBuffer** & **putDouble** (double value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

**Parameters**

value	The value to be written.
-------	--------------------------

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 964)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only

- virtual **ByteBuffer** & **putDouble** (int index, double value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes eight bytes containing the given value, into this buffer at the given index.*

**Parameters**

index	The position in the <b>Buffer</b> (p. 936) to write the data
value	The value to write.

**Returns**

*a reference to this buffer.*

**Exceptions**

IndexOutOfBoundsException	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only.

- virtual **ByteBuffer** & **putFloat** (float value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

**Parameters**

value	<i>The value to be written.</i>
-------	---------------------------------

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 964)	<i>if there are fewer than bytes remaining in this buffer than the size of the data to be written.</i>
<b>ReadOnlyBufferException</b> (p. 3260)	<i>if this buffer is read-only.</i>

- virtual **ByteBuffer & putFloat** (int index, float value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Writes four bytes containing the given value, into this buffer at the given index.*

**Parameters**

index	<i>The position in the <b>Buffer</b> (p. 936) to write the data</i>
value	<i>The value to write.</i>

**Returns**

*a reference to this buffer.*

**Exceptions**

IndexOutOfBoundsException	<i>if index greater than the buffer's limit minus the size of the type being written, or index is negative.</i>
<b>ReadOnlyBufferException</b> (p. 3260)	<i>if this buffer is read-only.</i>

- virtual **ByteBuffer & putLong** (long long value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

**Parameters**

value	<i>The value to be written.</i>
-------	---------------------------------

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 964)	<i>if there are fewer than bytes remaining in this buffer than the size of the data to be written.</i>
<b>ReadOnlyBufferException</b> (p. 3260)	<i>if this buffer is read-only.</i>

- virtual **ByteBuffer** & **putLong** (int index, long long value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes eight bytes containing the given value, into this buffer at the given index.*

**Parameters**

index	<i>The position in the <b>Buffer</b> (p. 936) to write the data.</i>
value	<i>The value to write.</i>

**Returns**

*a reference to this buffer.*

**Exceptions**

IndexOutOfBoundsException	<i>if index greater than the buffer's limit minus the size of the type being written, or index is negative.</i>
<b>ReadOnlyBufferException</b> (p. 3260)	<i>if this buffer is read-only.</i>

- virtual **ByteBuffer** & **putInt** (int value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

**Parameters**

value	<i>The value to be written.</i>
-------	---------------------------------

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 964)	<i>if there are fewer than bytes remaining in this buffer than the size of the data to be written</i>
<b>ReadOnlyBufferException</b> (p. 3260)	<i>if this buffer is read-only</i>

- virtual **ByteBuffer** & **putInt** (int index, int value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes four bytes containing the given value, into this buffer at the given index.*

**Parameters**

index	<i>The position in the <b>Buffer</b> (p. 936) to write the data.</i>
value	<i>The value to write.</i>

**Returns**

*a reference to this buffer*

**Exceptions**

IndexOutOfBoundsException	<i>if index greater than the buffer's limit minus the size of the type being written, or index is negative.</i>
---------------------------	---

<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only
---	-----------------------------

- virtual **ByteBuffer** & **putShort** (short value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

**Parameters**

value	The value to be written.
-------	--------------------------

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 964)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only.

- virtual **ByteBuffer** & **putShort** (int index, short value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes two bytes containing the given value, into this buffer at the given index.*

**Parameters**

index	The position in the <b>Buffer</b> (p. 936) to write the data
value	The value to write.

**Returns**

*a reference to this buffer*

**Exceptions**

<b>IndexOutOfBoundsException</b>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only.

- virtual **ByteBuffer** \* **slice** () const

*Creates a new byte buffer whose content is a shared subsequence of this buffer's content.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be*



*read-only if, and only if, this buffer is read-only.*

**Returns**

*the newly create **ByteBuffer** (p. 1049) which the caller owns.*

**Protected Member Functions**

- virtual void **setReadOnly** (bool value)

*Sets this **ByteBuffer** (p. 1004) as Read-Only or not Read-Only.*

**6.166.1 Detailed Description**

This class defines six categories of operations upon byte buffers: 1. Absolute and relative get and put methods that read and write single bytes; 2. Relative bulk get methods that transfer contiguous sequences of bytes from this buffer into an array; 3. Relative bulk put methods that transfer contiguous sequences of bytes from a byte array or some other byte buffer into this buffer; 4. Absolute and relative get and put methods that read and write values of other primitive types, translating them to and from sequences of bytes in a particular byte order; 5. Methods for creating view buffers, which allow a byte buffer to be viewed as a buffer containing values of some other primitive type; and 6. Methods for compacting, duplicating, and slicing a byte buffer.

Byte buffers can be created either by allocation, which allocates space for the buffer's content, or by wrapping an existing byte array into a buffer.

Access to binary data:

This class defines methods for reading and writing values of all other primitive types, except boolean. Primitive values are translated to (or from) sequences of bytes according to the buffer's current byte order.

For access to heterogeneous binary data, that is, sequences of values of different types, this class defines a family of absolute and relative get and put methods for each type. For 32-bit floating-point values, for example, this class defines:

float **getFloat**() (p. 1027) float **getFloat**(int index) void **putFloat**(float f) (p. 1034) void **putFloat**(int index, float f) (p. 1033)

Corresponding methods are defined for the types char, short, int, long, and double. The index parameters of the absolute get and put methods are in terms of bytes rather than of the type being read or written.

For access to homogeneous binary data, that is, sequences of values of the same type, this class defines methods that can create views of a given byte buffer. A view buffer is simply another buffer whose content is backed by the byte buffer. Changes to the byte buffer's content will be visible in the view buffer, and vice versa; the two buffers' position, limit, and mark values are independent. The **asFloatBuffer** method, for example, creates an instance of the **FloatBuffer** class that is backed by the byte buffer upon which the method is invoked. Corresponding view-creation methods are defined for the types char, short, int, long, and double.

View buffers have two important advantages over the families of type-specific get and put methods described above:

A view buffer is indexed not in terms of bytes but rather in terms of the type-specific size of its values;

A view buffer provides relative bulk get and put methods that can transfer contiguous sequences of values between a buffer and an array or some other buffer of the same type; and

#### Since

1.0

### 6.166.2 Constructor & Destructor Documentation

**6.166.2.1** `decaf::internal::nio::ByteBuffer::ByteBuffer ( int capacity, bool readOnly = false ) throw ( decaf::lang::exceptions::IllegalArgumentException )`

Creates a **ByteBuffer** (p. 1004) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

#### Parameters

<i>capacity</i>	The size of the array, this is the limit we read and write to.
<i>readOnly</i>	Should this buffer be read-only, default as false

#### Exceptions

<i>IllegalArgumentException</i>	if the capacity value is negative.
---------------------------------	------------------------------------

**6.166.2.2** `decaf::internal::nio::ByteBuffer::ByteBuffer ( unsigned char * array, int size, int offset, int length, bool readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a **ByteBuffer** (p. 1004) object that wraps the given array.

#### Parameters

<i>array</i>	The array to wrap.
<i>size</i>	The size of the array passed.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The size of the sub-array, this is the limit we read and write to.
<i>readOnly</i>	Should this buffer be read-only, default as false.

**Exceptions**

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset and length are violated.

**6.166.2.3** `decaf::internal::nio::ByteBuffer::ByteBuffer ( const decaf::lang::Pointer< ByteArrayAdapter > & array, int offset, int length, bool readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **ByteBuffer** (p. 1004) will be that of the remaining capacity of the passed buffer.

**Parameters**

<i>array</i>	The ByteArrayAdapter to wrap
<i>offset</i>	The offset into array where the buffer starts
<i>length</i>	The length of the array we are wrapping or limit.
<i>readOnly</i>	Boolean indicating if this a readOnly buffer.

**Exceptions**

<i>NullPointerException</i>	if array is NULL
<i>IndexOutOfBoundsException</i>	if offset is greater than array capacity.

**6.166.2.4** `decaf::internal::nio::ByteBuffer::ByteBuffer ( const ByteBuffer & other )`

Create a **ByteBuffer** (p. 1004) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.

**Parameters**

<i>other</i>	The <b>ByteBuffer</b> (p. 1004) this one is to mirror.
--------------	--

6.166.2.5 `virtual decaf::internal::nio::ByteBuffer::~~ByteBuffer ( ) [virtual]`

### 6.166.3 Member Function Documentation

6.166.3.1 `virtual unsigned char* decaf::internal::nio::ByteBuffer::array  
( ) throw ( decaf::nio::ReadOnlyBufferException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Returns the byte array that backs this buffer.

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

The array that backs this buffer

#### Exceptions

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this buffer is backed by an array but is read-only
<i><b>UnsupportedOperationException</b></i>	if this buffer is not backed by an accessible array

Implements `decaf::nio::ByteBuffer` (p. 1056).

6.166.3.2 `virtual int decaf::internal::nio::ByteBuffer::arrayOffset ( )  
const throw ( decaf::nio::ReadOnlyBufferException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer.

If this buffer is backed by an array then buffer position `p` corresponds to array index `p + arrayOffset()` (p. 1057).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

The offset within this buffer's array of the first element of the buffer.

#### Exceptions

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this buffer is backed by an array but is read-only.
--	--

<i>UnsupportedOperationException</i>	if this buffer is not backed by an accessible array.
--------------------------------------	--

Implements **decaf::nio::ByteBuffer** (p. 1057).

**6.166.3.3** `virtual decaf::nio::CharBuffer* decaf::internal::nio::ByteBuffer::asCharBuffer ( ) const`  
`[inline, virtual]`

Creates a view of this byte buffer as a char buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the new Char **Buffer** (p. 936), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 1057).

**6.166.3.4** `virtual decaf::nio::DoubleBuffer* decaf::internal::nio::ByteBuffer::asDoubleBuffer ( ) const`  
`[inline, virtual]`

Creates a view of this byte buffer as a double buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the new double **Buffer** (p. 936), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 1058).

**6.166.3.5** `virtual decaf::nio::FloatBuffer* decaf::internal::nio::ByteBuffer::asFloatBuffer ( ) const`  
`[inline, virtual]`

Creates a view of this byte buffer as a float buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the new float **Buffer** (p. 936), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 1058).

```
6.166.3.6  virtual decaf::nio::IntBuffer* decaf::internal::nio::ByteBuffer::asIntBuffer (
            ) const [inline, virtual]
```

Creates a view of this byte buffer as a int buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the new int **Buffer** (p. 936), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 1058).

```
6.166.3.7  virtual decaf::nio::LongBuffer* de-
            caf::internal::nio::ByteBuffer::asLongBuffer (    ) const
            [inline, virtual]
```

Creates a view of this byte buffer as a long buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the new long **Buffer** (p. 936), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 1059).

**6.166.3.8** `virtual ByteBuffer* decaf::internal::nio::ByteBuffer::asReadOnlyBuffer ( ) const [virtual]`

Creates a new, read-only byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns

The new, read-only byte buffer which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 1059).

**6.166.3.9** `virtual decaf::nio::ShortBuffer* decaf::internal::nio::ByteBuffer::asShortBuffer ( ) const [inline, virtual]`

Creates a view of this byte buffer as a short buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the new short **Buffer** (p. 936), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 1059).

**6.166.3.10** `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::compact ( ) throw ( decaf::nio::ReadOnlyBufferException ) [virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 941) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 940) - 1 is copied to index  $n = \text{limit}()$  (p. 940) - 1 -  $p$ . The buffer's

position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

### Returns

a reference to this **ByteBuffer** (p. 1049).

### Exceptions

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this buffer is read-only.
--	------------------------------

Implements **decaf::nio::ByteBuffer** (p. 1060).

**6.166.3.11** `virtual ByteBuffer* decaf::internal::nio::ByteBuffer::duplicate ( )`  
[virtual]

Creates a new byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

a new **Byte Buffer** (p. 936) which the caller owns.

Implements **decaf::nio::ByteBuffer** (p. 1060).

**6.166.3.12** `virtual unsigned char decaf::internal::nio::ByteBuffer::get ( ) const throw ( decaf::nio::BufferUnderflowException )` [virtual]

Relative get method.

Reads the byte at this buffer's current position, and then increments the position.

### Returns

The byte at the buffer's current position.

### Exceptions

<i><b>BufferUnderflowException</b></i> (p. 967)	if the buffer's current position is not smaller than its limit.
--	---



Implements **decaf::nio::ByteBuffer** (p. 1061).

**6.166.3.13** virtual unsigned char decaf::internal::nio::ByteBuffer::get ( int *index* )  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Absolute get method.

Reads the byte at the given index.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the byte is to be read.
--------------	---

#### Returns

the byte that is located at the given index.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implements **decaf::nio::ByteBuffer** (p. 1062).

**6.166.3.14** virtual char decaf::internal::nio::ByteBuffer::getChar ( ) throw ( decaf::nio::BufferUnderflowException ) [inline, virtual]

Reads the next byte at this buffer's current position, and then increments the position by one.

#### Returns

the next char in the buffer.

#### Exceptions

<i>BufferUnderflowException</i> (p. 967)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

Implements **decaf::nio::ByteBuffer** (p. 1063).

**6.166.3.15** virtual char decaf::internal::nio::ByteBuffer::getChar ( int *index* ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [inline, virtual]

Reads one byte at the given index and returns it.

**Parameters**

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the byte is to be read.
--------------	---

**Returns**

the char at the given index in the buffer

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implements **decaf::nio::ByteBuffer** (p. 1063).

**6.166.3.16** `virtual double decaf::internal::nio::ByteBuffer::getDouble ( ) throw ( decaf::nio::BufferUnderflowException ) [virtual]`

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

**Returns**

the next double in the buffer.

**Exceptions**

<i>BufferUnderflowException</i> (p. 967)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

Implements **decaf::nio::ByteBuffer** (p. 1064).

**6.166.3.17** `virtual double decaf::internal::nio::ByteBuffer::getDouble ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Reads eight bytes at the given index and returns it.

**Parameters**

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the bytes are to be read.
--------------	---

**Returns**

the double at the given index in the buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implements **decaf::nio::ByteBuffer** (p. 1064).

**6.166.3.18** virtual float decaf::internal::nio::ByteBuffer::getFloat ( ) throw ( decaf::nio::BufferUnderflowException ) [virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

#### Returns

the next float in the buffer.

#### Exceptions

<i>BufferUnderflowException</i> (p. 967)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

Implements **decaf::nio::ByteBuffer** (p. 1064).

**6.166.3.19** virtual float decaf::internal::nio::ByteBuffer::getFloat ( int *index* ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Reads four bytes at the given index and returns it.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the bytes are to be read.
--------------	---

#### Returns

the float at the given index in the buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

Implements **decaf::nio::ByteBuffer** (p. 1065).

**6.166.3.20** virtual int decaf::internal::nio::ByteBuffer::getInt ( ) throw ( decaf::nio::BufferUnderflowException ) [virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

#### Returns

the next int in the buffer.

**Exceptions**

<i><b>BufferUnderflowException</b></i> (p. 967)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
--	---

Implements **decaf::nio::ByteBuffer** (p. 1065).

**6.166.3.21** `virtual int decaf::internal::nio::ByteBuffer::getInt ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Reads four bytes at the given index and returns it.

**Parameters**

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the bytes are to be read.
--------------	---

**Returns**

the int at the given index in the buffer.

**Exceptions**

<i><b>IndexOutOfBoundsException</b></i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
---	--

Implements **decaf::nio::ByteBuffer** (p. 1066).

**6.166.3.22** `virtual long long decaf::internal::nio::ByteBuffer::getLong ( ) throw ( decaf::nio::BufferUnderflowException )` [virtual]

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

**Returns**

the next long long in the buffer.

**Exceptions**

<i><b>BufferUnderflowException</b></i> (p. 967)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
--	---

Implements **decaf::nio::ByteBuffer** (p. 1066).

**6.166.3.23** `virtual long long decaf::internal::nio::ByteBuffer::getLong ( int index )  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]`

Reads eight bytes at the given index and returns it.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the bytes are to be read.
--------------	---

#### Returns

the long long at the given index in the buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

Implements **decaf::nio::ByteBuffer** (p. 1066).

**6.166.3.24** `virtual short decaf::internal::nio::ByteBuffer::getShort ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Reads two bytes at the given index and returns it.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the bytes are to be read.
--------------	---

#### Returns

the short at the given index in the buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

Implements **decaf::nio::ByteBuffer** (p. 1067).

**6.166.3.25** `virtual short decaf::internal::nio::ByteBuffer::getShort ( ) throw ( decaf::nio::BufferUnderflowException ) [virtual]`

Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

#### Returns

the next short in the buffer.

**Exceptions**

<b><i>BufferUnderflowException</i></b> (p. 967)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
--	---

Implements **decaf::nio::ByteBuffer** (p. 1067).

**6.166.3.26** `virtual bool decaf::internal::nio::ByteBuffer::hasArray ( ) const` [inline, virtual]

Tells whether or not this buffer is backed by an accessible byte array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns**

true if, and only if, this buffer is backed by an array and is not read-only.

Implements **decaf::nio::ByteBuffer** (p. 1068).

**6.166.3.27** `virtual bool decaf::internal::nio::ByteBuffer::isReadOnly ( ) const` [inline, virtual]

Tells whether or not this buffer is read-only.

**Returns**

true if, and only if, this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 1068).

**6.166.3.28** `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::put ( unsigned char value ) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )` [virtual]

Writes the given byte into this buffer at the current position, and then increments the position.

**Parameters**

<i>value</i>	- the byte value to be written.
--------------	---------------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 964)	if this buffer's current position is not smaller than its limit.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 1068).

**6.166.3.29** virtual **ByteBuffer&** decaf::internal::nio::ByteBuffer::put  
( int *index*, unsigned char *value* ) throw ( **decaf::lang::exceptions::IndexOutOfBoundsException**,  
**decaf::nio::ReadOnlyBufferException** ) [virtual]

Writes the given byte into this buffer at the given index.

**Parameters**

<i>index</i>	- position in the <b>Buffer</b> (p. 936) to write the data
<i>value</i>	- the byte to write.

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>IndexOutOfBoundsException</i></b>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 1069).

**6.166.3.30** virtual **ByteBuffer&** decaf::internal::nio::ByteBuffer::putChar  
( int *index*, char *value* ) throw ( **decaf::lang::exceptions::IndexOutOfBoundsException**,  
**decaf::nio::ReadOnlyBufferException** ) [virtual]

Writes one byte containing the given value, into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data.
<i>value</i>	The value to write.

**Returns**

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<i>ReadOnlyBufferException</i> (p. 3260)	if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 1072).

6.166.3.31 **virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putChar**  
( *char value* ) throw ( **decaf::nio::BufferOverflowException**,  
**decaf::nio::ReadOnlyBufferException** ) [virtual]

Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

**Parameters**

<i>value</i>	The value to be written.
--------------	--------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<i>BufferOverflowException</i> (p. 964)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
<i>ReadOnlyBufferException</i> (p. 3260)	if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 1071).

6.166.3.32 **virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putDouble**  
( *int index*, *double value* ) throw ( **decaf::lang::exceptions::IndexOutOfBoundsException**,  
**decaf::nio::ReadOnlyBufferException** ) [virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data
<i>value</i>	The value to write.



**Returns**

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<i>ReadOnlyBufferException</i> (p. 3260)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 1073).

**6.166.3.33** virtual **ByteBuffer&** decaf::internal::nio::ByteBuffer::putDouble  
( double *value* ) throw ( decaf::nio::BufferOverflowException,  
decaf::nio::ReadOnlyBufferException ) [virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters**

<i>value</i>	The value to be written.
--------------	--------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<i>BufferOverflowException</i> (p. 964)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
<i>ReadOnlyBufferException</i> (p. 3260)	if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 1072).

**6.166.3.34** virtual **ByteBuffer&** decaf::internal::nio::ByteBuffer::putFloat  
( int *index*, float *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::nio::ReadOnlyBufferException ) [virtual]

Writes four bytes containing the given value, into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data
<i>value</i>	The value to write.

**Returns**

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<i>ReadOnlyBufferException</i> (p. 3260)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 1074).

6.166.3.35 **virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putFloat ( float *value* ) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )** [virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters**

<i>value</i>	The value to be written.
--------------	--------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<i>BufferOverflowException</i> (p. 964)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
<i>ReadOnlyBufferException</i> (p. 3260)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 1073).

6.166.3.36 **virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putInt ( int *index*, int *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )** [virtual]

Writes four bytes containing the given value, into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data.
<i>value</i>	The value to write.

**Returns**

a reference to this buffer

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<i>ReadOnlyBufferException</i> (p. 3260)	if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 1074).

**6.166.3.37** virtual **ByteBuffer&** decaf::internal::nio::ByteBuffer::putInt  
( int *value* ) throw ( decaf::nio::BufferOverflowException,  
decaf::nio::ReadOnlyBufferException ) [virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters**

<i>value</i>	The value to be written.
--------------	--------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<i>BufferOverflowException</i> (p. 964)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
<i>ReadOnlyBufferException</i> (p. 3260)	if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 1075).

**6.166.3.38** virtual **ByteBuffer&** decaf::internal::nio::ByteBuffer::putLong  
( long long *value* ) throw ( decaf::nio::BufferOverflowException,  
decaf::nio::ReadOnlyBufferException ) [virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters**

<i>value</i>	The value to be written.
--------------	--------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<i><b>BufferOverflowException</b></i> (p. 964)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 1075).

**6.166.3.39** virtual **ByteBuffer&** **decaf::internal::nio::ByteBuffer::putLong**  
( int *index*, long long *value* ) throw ( **decaf::lang::exceptions::IndexOutOfBoundsException**,  
**decaf::nio::ReadOnlyBufferException** ) [virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data.
<i>value</i>	The value to write.

**Returns**

a reference to this buffer.

**Exceptions**

<i><b>IndexOutOfBoundsException</b></i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 1076).

**6.166.3.40** virtual **ByteBuffer&** **decaf::internal::nio::ByteBuffer::putShort**  
( short *value* ) throw ( **decaf::nio::BufferOverflowException**,  
**decaf::nio::ReadOnlyBufferException** ) [virtual]

Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters**

<i>value</i>	The value to be written.
--------------	--------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 964)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 1077).

**6.166.3.41** virtual **ByteBuffer&** decaf::internal::nio::ByteBuffer::putShort ( int *index*, short *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException ) [virtual]

Writes two bytes containing the given value, into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data
<i>value</i>	The value to write.

**Returns**

a reference to this buffer

**Exceptions**

<b><i>IndexOutOfBoundsException</i></b>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 1076).

**6.166.3.42** virtual void decaf::internal::nio::ByteBuffer::setReadOnly ( bool *value* ) [inline, protected, virtual]

Sets this **ByteBuffer** (p. 1004) as Read-Only or not Read-Only.

**Parameters**

<i>value</i>	Boolean value, true if this buffer is to be read-only, false otherwise.
--------------	---

**6.166.3.43** `virtual ByteBuffer* decaf::internal::nio::ByteBuffer::slice ( ) const`  
`[virtual]`

Creates a new byte buffer whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the newly create **ByteBuffer** (p. 1049) which the caller owns.

Implements **decaf::nio::ByteBuffer** (p. 1077).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/ByteBuffer.h`

## 6.167 decaf::io::ByteArrayInputStream Class Reference

A **ByteArrayInputStream** (p. 1038) contains an internal buffer that contains bytes that may be read from the stream.

```
#include <src/main/decaf/io/ByteArrayInputStream.h>
```

Inheritance diagram for `decaf::io::ByteArrayInputStream`:

### Public Member Functions

- **ByteArrayInputStream ()**

*Creates an **ByteArrayInputStream** (p. 1038) with an empty input buffer; the buffer can be initialized with a call to `setByteArray`.*

- **ByteArrayInputStream (const std::vector< unsigned char > &buffer)**

*Creates the input stream and calls `setBuffer` with the specified buffer object.*

- **ByteArrayInputStream (const unsigned char \*buffer, int bufferSize, bool own=false)**

`throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )`

*Create an instance of the **ByteArrayInputStream** (p. 1038) with the given buffer as the source of input for all read operations.*

- **ByteArrayInputStream** (const unsigned char \*buffer, int bufferSize, int offset, int length, bool own=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )

Create an instance of the **ByteArrayInputStream** (p. 1038) with the given buffer as the source of input for all read operations.

- virtual ~**ByteArrayInputStream** ()
- virtual void **setByteArray** (const std::vector< unsigned char > &buffer)
- virtual void **setByteArray** (const unsigned char \*buffer, int bufferSize) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )

Sets the internal buffer.

Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.

- virtual void **setByteArray** (const unsigned char \*buffer, int bufferSize, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )

Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.

- virtual int **available** () const throw ( IOException )

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

#### Returns

the number of bytes available on this input stream.

#### Exceptions

<b>IOException</b> (p. 2210)	if an I/O error occurs.
------------------------------	-------------------------

- virtual long long **skip** (long long num) throw ( io::IOException, lang::exceptions::UnsupportedOperationException )

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2105) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters

num	The number of bytes to skip.
-----	------------------------------

**Returns**

*total bytes skipped*

**Exceptions**

<b>IOException</b> (p. 2210)	<i>if an I/O error occurs.</i>
<b>UnsupportedOperationException</b>	<i>if the concrete stream class does not support skipping bytes.</i>

- virtual void **mark** (int readLimit)

*Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.*

*If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.*

*Calling mark on a closed stream instance should have no effect.*

*The default implementation of this method does nothing.*

**Parameters**

readLimit	<i>The max bytes read before marked position is invalid.</i>
-----------	--

- virtual void **reset** () throw ( IOException )

*Repositions this stream to the position at the time the mark method was last called on this input stream.*

*If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2210) might be thrown. \* If such an **IOException** (p. 2210) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.*

*If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2210). \* If an **IOException** (p. 2210) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.*

*The default implementation of this method throws an **IOException** (p. 2210).*

**Exceptions**

<b>IOException</b> (p. 2210)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

- virtual bool **markSupported** () const

*Determines if this input stream supports the mark and reset methods.*

*Whether or not mark and reset are supported is an invariant property of a particular input stream instance.*

*The default implementation of this method returns false.*

**Returns**

*true if this stream instance supports marks*



## Protected Member Functions

- virtual int **doReadByte** () throw ( IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

### 6.167.1 Detailed Description

A **ByteArrayInputStream** (p. 1038) contains an internal buffer that contains bytes that may be read from the stream. An internal counter keeps track of the next byte to be supplied by the read method. The **ByteArrayInputStream** (p. 1038) never copies the supplied buffers, only points to them, therefore the caller must ensure that the supplied buffer remain in scope, or is not deleted before this **ByteArrayInputStream** (p. 1038) is freed. If the own argument of one of the constructors that accepts an array pointer is set to true than the **ByteArrayInputStream** (p. 1038) instance will take ownership of the supplied pointer and delete it when that instance is destroyed.

Closing a **ByteArrayInputStream** (p. 1038) has no effect. The methods in this class can be called after the stream has been closed without generating an **IOException** (p. 2210).

Since

1.0

### 6.167.2 Constructor & Destructor Documentation

#### 6.167.2.1 decaf::io::ByteArrayInputStream::ByteArrayInputStream ( )

Creates an **ByteArrayInputStream** (p. 1038) with an empty input buffer, the buffer can be initialized with a call to **setByteArray**.

#### 6.167.2.2 decaf::io::ByteArrayInputStream::ByteArrayInputStream ( const std::vector< unsigned char > & buffer )

Creates the input stream and calls **setBuffer** with the specified buffer object.

#### Parameters

<i>buffer</i>	The buffer to be used.
---------------	------------------------

```
6.167.2.3 decaf::io::ByteArrayInputStream::ByteArrayInputStream ( const
unsigned char * buffer, int bufferSize, bool own = false )
throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException )
```

Create an instance of the **ByteArrayInputStream** (p. 1038) with the given buffer as the source of input for all read operations.

#### Parameters

<i>buffer</i>	The initial byte array to use to read from.
<i>bufferSize</i>	The size of the buffer.
<i>own</i>	Indicates if this object should take ownership of the array, default is false.

#### Exceptions

<i>NullPointerException</i>	if the buffer is Null.
<i>IllegalArgumentEx-ception</i>	if the bufferSize is negative.

```
6.167.2.4 decaf::io::ByteArrayInputStream::ByteArrayInputStream ( const unsigned
char * buffer, int bufferSize, int offset, int length, bool own = false
) throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException )
```

Create an instance of the **ByteArrayInputStream** (p. 1038) with the given buffer as the source of input for all read operations.

#### Parameters

<i>buffer</i>	The initial byte array to use to read from.
<i>bufferSize</i>	The size of the buffer.
<i>offset</i>	The offset into the buffer to begin reading from.
<i>length</i>	The number of bytes to read past the offset.
<i>own</i>	Indicates if this object should take ownership of the array, default is false.

#### Exceptions

<i>NullPointerException</i>	if the buffer is Null.
<i>IllegalArgumentEx-ception</i>	if the bufferSize is negative.

6.167.2.5 `virtual decaf::io::ByteArrayInputStream::~~ByteArrayInputStream ( )`  
[virtual]

### 6.167.3 Member Function Documentation

6.167.3.1 `virtual int decaf::io::ByteArrayInputStream::available ( ) const throw ( IOException )` [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

#### Returns

the number of bytes available on this input stream.

#### Exceptions

<i>IOException</i> (p. 2210)	if an I/O error occurs.
---------------------------------	-------------------------

Reimplemented from **decaf::io::InputStream** (p. 2107).

6.167.3.2 `virtual int decaf::io::ByteArrayInputStream::doReadArrayBounded ( unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )` [protected, virtual]

Reimplemented from **decaf::io::InputStream** (p. 2108).

6.167.3.3 `virtual int decaf::io::ByteArrayInputStream::doReadByte ( ) throw ( IOException )` [protected, virtual]

Implements **decaf::io::InputStream** (p. 2109).

6.167.3.4 `virtual void decaf::io::ByteArrayInputStream::mark ( int readLimit )` [virtual]

Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit

is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

#### Parameters

<i>readLimit</i>	The max bytes read before marked position is invalid.
------------------	---

Reimplemented from **decaf::io::InputStream** (p. 2109).

**6.167.3.5** `virtual bool decaf::io::ByteArrayInputStream::markSupported ( ) const`  
`[inline, virtual]`

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

#### Returns

true if this stream instance supports marks

Reimplemented from **decaf::io::InputStream** (p. 2109).

**6.167.3.6** `virtual void decaf::io::ByteArrayInputStream::reset ( ) throw ( IOException )`  
`[virtual]`

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2210) might be thrown. \* If such an **IOException** (p. 2210) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2210). \* If an **IOException** (p. 2210) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2210).

#### Exceptions

<i>IOException</i> (p. 2210)	if an I/O error occurs.
---------------------------------	-------------------------

Reimplemented from **decaf::io::InputStream** (p. 2113).

**6.167.3.7** `virtual void decaf::io::ByteArrayInputStream::setByteArray ( const unsigned char *  
buffer, int bufferSize ) throw ( decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IllegalArgumentException ) [virtual]`

Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.

#### Parameters

<i>buffer</i>	The initial byte array to use to read from.
<i>bufferSize</i>	The size of the buffer.

#### Exceptions

<i>NullPointerException</i>	if the buffer is Null.
<i>IllegalArgumentEx- ception</i>	if the bufferSize is negative.

**6.167.3.8** `virtual void decaf::io::ByteArrayInputStream::setByteArray ( const std::vector<  
unsigned char > &buffer ) [virtual]`

Sets the internal buffer.

The input stream will wrap around this buffer and will perform all read operations on it. The position will be reinitialized to the beginning of the specified buffer. This class will not own the given buffer - it is the caller's responsibility to free the memory of the given buffer as appropriate.

#### Parameters

<i>buffer</i>	The buffer to be used.
---------------	------------------------

**6.167.3.9** `virtual void decaf::io::ByteArrayInputStream::setByteArray ( const  
unsigned char * buffer, int bufferSize, int offset, int length )  
throw ( decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IllegalArgumentException ) [virtual]`

Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.

**Parameters**

<i>buffer</i>	The initial byte array to use to read from.
<i>bufferSize</i>	The size of the buffer.
<i>offset</i>	The offset into the buffer to begin reading from.
<i>length</i>	The number of bytes to read past the offset.

**Exceptions**

<i>NullPointerException</i>	if the buffer is Null.
<i>IllegalArgumentEx- ception</i>	if the bufferSize is negative.

**6.167.3.10** `virtual long long decaf::io::ByteArrayInputStream::skip ( long long num ) throw ( io::IOException, lang::exceptions::UnsupportedOperationException )`  
[virtual]

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2105) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

**Parameters**

<i>num</i>	The number of bytes to skip.
------------	------------------------------

**Returns**

total bytes skipped

**Exceptions**

<i>IOException</i> (p. 2210)	if an I/O error occurs.
<i>UnsupportedOperationEx- ception</i>	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 2113).

The documentation for this class was generated from the following file:

- src/main/decaf/io/ByteArrayInputStream.h

## 6.168 decaf::io::ByteArrayOutputStream Class Reference

```
#include <src/main/decaf/io/ByteArrayOutputStream.h>
```

Inheritance diagram for decaf::io::ByteArrayOutputStream:

### Public Member Functions

- **ByteArrayOutputStream** ()  
*Default Constructor - uses a default internal buffer of 32 bytes, the size increases as the need for more room arises.*
- **ByteArrayOutputStream** (int bufferSize) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **ByteArrayOutputStream** (p. 1046) with an internal buffer allocated with the given size.*
- virtual ~**ByteArrayOutputStream** ()
- std::pair< unsigned char \*, int > **toByteArray** () const  
*Creates a newly allocated byte array.*
- long long **size** () const  
*Gets the current count of bytes written into this **ByteArrayOutputStream** (p. 1046).*
- virtual void **reset** () throw ( IOException )  
*Clear current Stream contents.*
- virtual std::string **toString** () const  
*Converts the bytes in the buffer into a standard C++ string.*
- void **writeTo** (**OutputStream** \*out) const throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )  
*Writes the complete contents of this byte array output stream to the specified output stream argument, as if by calling the output stream's write method using out.write(buf, 0, count ).*

### Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### 6.168.1 Constructor & Destructor Documentation

#### 6.168.1.1 `decaf::io::ByteArrayOutputStream::ByteArrayOutputStream ( )`

Default Constructor - uses a default internal buffer of 32 bytes, the size increases as the need for more room arises.

#### 6.168.1.2 `decaf::io::ByteArrayOutputStream::ByteArrayOutputStream ( int bufferSize ) throw ( decaf::lang::exceptions::IllegalArgumentException )`

Creates a **ByteArrayOutputStream** (p. 1046) with an internal buffer allocated with the given size.

##### Parameters

<i>bufferSize</i>	The size to use for the internal buffer.
-------------------	--

##### Exceptions

<i>IllegalArgumentException</i>	if the size is less than or equal to zero.
---------------------------------	--

#### 6.168.1.3 `virtual decaf::io::ByteArrayOutputStream::~~ByteArrayOutputStream ( )` [virtual]

### 6.168.2 Member Function Documentation

#### 6.168.2.1 `virtual void decaf::io::ByteArrayOutputStream::doWriteArrayBounded ( const unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [protected, virtual]

Reimplemented from **decaf::io::OutputStream** (p. 2994).

#### 6.168.2.2 `virtual void decaf::io::ByteArrayOutputStream::doWriteByte ( unsigned char value ) throw ( decaf::io::IOException )` [protected, virtual]

Implements **decaf::io::OutputStream** (p. 2994).

#### 6.168.2.3 `virtual void decaf::io::ByteArrayOutputStream::reset ( ) throw ( IOException )` [virtual]

Clear current Stream contents.

##### Exceptions



<i><b>IOException</b></i> (p. 2210)	
--	--

#### 6.168.2.4 long long decaf::io::ByteArrayOutputStream::size ( ) const

Gets the current count of bytes written into this **ByteArrayOutputStream** (p. 1046).

##### Returns

the number of valid bytes contained in the **ByteArrayOutputStream** (p. 1046).

#### 6.168.2.5 std::pair<unsigned char\*, int> decaf::io::ByteArrayOutputStream::toByteArray ( ) const

Creates a newly allocated byte array.

Its size is the current size of this output stream and the valid contents of the buffer have been copied into it. The newly allocated array and its size are returned inside an STL pair structure, the caller is responsible for freeing the returned array.

##### Returns

an STL pair containing the copied array and its size.

#### 6.168.2.6 virtual std::string decaf::io::ByteArrayOutputStream::toString ( ) const [virtual]

Converts the bytes in the buffer into a standard C++ string.

##### Returns

a string containing the bytes in the buffer

Reimplemented from **decaf::io::OutputStream** (p. 2995).

#### 6.168.2.7 void decaf::io::ByteArrayOutputStream::writeTo( OutputStream \* out ) const throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )

Writes the complete contents of this byte array output stream to the specified output stream argument, as if by calling the output stream's write method using out.write( buf, 0, count ).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**ByteArrayOutputStream.h**

## 6.169 decaf::nio::ByteBuffer Class Reference

This class defines six categories of operations upon byte buffers:

```
#include <src/main/decaf/nio/ByteBuffer.h>
```

Inheritance diagram for decaf::nio::ByteBuffer:

### Public Member Functions

- virtual **~ByteBuffer** ()
- virtual std::string **toString** () const
- **ByteBuffer & get** (std::vector< unsigned char > buffer) throw ( BufferUnderflowException )  
*Relative bulk get method.*
- **ByteBuffer & get** (unsigned char \*buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Relative bulk get method.*
- **ByteBuffer & put** (ByteBuffer &src) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )  
*This method transfers the bytes remaining in the given source buffer into this buffer.*
- **ByteBuffer & put** (const unsigned char \*buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*This method transfers bytes into this buffer from the given source array.*
- **ByteBuffer & put** (std::vector< unsigned char > &buffer) throw ( BufferOverflowException, ReadOnlyBufferException )  
*This method transfers the entire content of the given source byte array into this buffer.*
- virtual bool **isReadOnly** () const =0  
*Tells whether or not this buffer is read-only.*
- virtual unsigned char \* **array** ()=0 throw ( ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException )  
*Returns the byte array that backs this buffer.*
- virtual int **arrayOffset** () const =0 throw ( ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException )  
*Returns the offset within this buffer's backing array of the first element of the buffer.*
- virtual bool **hasArray** () const =0

*Tells whether or not this buffer is backed by an accessible byte array.*

- virtual **CharBuffer** \* **asCharBuffer** () const =0  
*Creates a view of this byte buffer as a char buffer.*
- virtual **DoubleBuffer** \* **asDoubleBuffer** () const =0  
*Creates a view of this byte buffer as a double buffer.*
- virtual **FloatBuffer** \* **asFloatBuffer** () const =0  
*Creates a view of this byte buffer as a float buffer.*
- virtual **IntBuffer** \* **asIntBuffer** () const =0  
*Creates a view of this byte buffer as a int buffer.*
- virtual **LongBuffer** \* **asLongBuffer** () const =0  
*Creates a view of this byte buffer as a long buffer.*
- virtual **ShortBuffer** \* **asShortBuffer** () const =0  
*Creates a view of this byte buffer as a short buffer.*
- virtual **ByteBuffer** \* **asReadOnlyBuffer** () const =0  
*Creates a new, read-only byte buffer that shares this buffer's content.*
- virtual **ByteBuffer** & **compact** ()=0 throw ( **ReadOnlyBufferException** )  
*Compacts this buffer.*
- virtual **ByteBuffer** \* **duplicate** ()=0  
*Creates a new byte buffer that shares this buffer's content.*
- virtual unsigned char **get** () const =0 throw ( **BufferUnderflowException** )  
*Relative get method.*
- virtual unsigned char **get** (int index) const =0 throw ( **decaf::lang::exceptions::IndexOutOfBoundsException** )  
*Absolute get method.*
- virtual char **getChar** ()=0 throw ( **BufferUnderflowException** )  
*Reads the next byte at this buffer's current position, and then increments the position by one.*
- virtual char **getChar** (int index) const =0 throw ( **decaf::lang::exceptions::IndexOutOfBoundsException** )  
*Reads one byte at the given index and returns it.*
- virtual double **getDouble** ()=0 throw ( **BufferUnderflowException** )  
*Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.*

- virtual double **getDouble** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads eight bytes at the given index and returns it.*
- virtual float **getFloat** ()=0 throw ( BufferUnderflowException )  
*Reads the next four bytes at this buffer's current position, and then increments the position by that amount.*
- virtual float **getFloat** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads four bytes at the given index and returns it.*
- virtual long long **getLong** ()=0 throw ( BufferUnderflowException )  
*Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.*
- virtual long long **getLong** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads eight bytes at the given index and returns it.*
- virtual int **getInt** ()=0 throw ( BufferUnderflowException )  
*Reads the next four bytes at this buffer's current position, and then increments the position by that amount.*
- virtual int **getInt** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads four bytes at the given index and returns it.*
- virtual short **getShort** ()=0 throw ( BufferUnderflowException )  
*Reads the next two bytes at this buffer's current position, and then increments the position by that amount.*
- virtual short **getShort** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads two bytes at the given index and returns it.*
- virtual **ByteBuffer & put** (unsigned char value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )  
*Writes the given byte into this buffer at the current position, and then increments the position.*
- virtual **ByteBuffer & put** (int index, unsigned char value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )  
*Writes the given byte into this buffer at the given index.*
- virtual **ByteBuffer & putChar** (char value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )

*Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.*

- virtual **ByteBuffer & putChar** (int index, char value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )

*Writes one byte containing the given value, into this buffer at the given index.*

- virtual **ByteBuffer & putDouble** (double value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )

*Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

- virtual **ByteBuffer & putDouble** (int index, double value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )

*Writes eight bytes containing the given value, into this buffer at the given index.*

- virtual **ByteBuffer & putFloat** (float value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )

*Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

- virtual **ByteBuffer & putFloat** (int index, float value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )

*Writes four bytes containing the given value, into this buffer at the given index.*

- virtual **ByteBuffer & putLong** (long long value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )

*Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

- virtual **ByteBuffer & putLong** (int index, long long value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )

*Writes eight bytes containing the given value, into this buffer at the given index.*

- virtual **ByteBuffer & putInt** (int value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )

*Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

- virtual **ByteBuffer & putInt** (int index, int value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )

*Writes four bytes containing the given value, into this buffer at the given index.*

- virtual **ByteBuffer & putShort** (short value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )

*Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

- virtual **ByteBuffer** & **putShort** (int index, short value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Writes two bytes containing the given value, into this buffer at the given index.*

- virtual **ByteBuffer** \* **slice** () const =0

*Creates a new byte buffer whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **ByteBuffer** &value) const

- virtual bool **equals** (const **ByteBuffer** &value) const

- virtual bool **operator==** (const **ByteBuffer** &value) const

- virtual bool **operator<** (const **ByteBuffer** &value) const

### Static Public Member Functions

- static **ByteBuffer** \* **allocate** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.*

- static **ByteBuffer** \* **wrap** (unsigned char \*array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new **ByteBuffer** (p. 1049).*

- static **ByteBuffer** \* **wrap** (std::vector< unsigned char > &buffer)

*Wraps the passed STL Byte Vector in a **ByteBuffer** (p. 1049).*

### Protected Member Functions

- **ByteBuffer** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Creates a **ByteBuffer** (p. 1049) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

### 6.169.1 Detailed Description

This class defines six categories of operations upon byte buffers: 1. Absolute and relative get and put methods that read and write single bytes; 2. Relative bulk get methods that transfer contiguous sequences of bytes from this buffer into an array; 3. Relative bulk put methods that transfer contiguous sequences of bytes from a byte array or some other byte buffer into this buffer; 4. Absolute and relative get and put methods that read and write values of other primitive types, translating them to and from sequences of bytes in a particular byte order; 5. Methods for creating view buffers, which allow a byte buffer to be viewed as a buffer containing values of some other primitive type; and 6. Methods for compacting, duplicating, and slicing a byte buffer.

Byte buffers can be created either by allocation, which allocates space for the buffer's content, or by wrapping an existing byte array into a buffer.

Access to binary data:

This class defines methods for reading and writing values of all other primitive types, except boolean. Primitive values are translated to (or from) sequences of bytes according to the buffer's current byte order.

For access to heterogeneous binary data, that is, sequences of values of different types, this class defines a family of absolute and relative get and put methods for each type. For 32-bit floating-point values, for example, this class defines:

float **getFloat()** (p. 1064) float **getFloat(int index)** void **putFloat(float f)** (p. 1073) void **putFloat(int index, float f)** (p. 1074)

Corresponding methods are defined for the types char, short, int, long, and double. The index parameters of the absolute get and put methods are in terms of bytes rather than of the type being read or written.

For access to homogeneous binary data, that is, sequences of values of the same type, this class defines methods that can create views of a given byte buffer. A view buffer is simply another buffer whose content is backed by the byte buffer. Changes to the byte buffer's content will be visible in the view buffer, and vice versa; the two buffers' position, limit, and mark values are independent. The **asFloatBuffer** method, for example, creates an instance of the **FloatBuffer** (p. 1985) class that is backed by the byte buffer upon which the method is invoked. Corresponding view-creation methods are defined for the types char, short, int, long, and double.

View buffers have two important advantages over the families of type-specific get and put methods described above:

A view buffer is indexed not in terms of bytes but rather in terms of the type-specific size of its values;

A view buffer provides relative bulk get and put methods that can transfer contiguous sequences of values between a buffer and an array or some other buffer of the same type; and

## 6.169.2 Constructor & Destructor Documentation

6.169.2.1 `decaf::nio::ByteBuffer::ByteBuffer ( int capacity ) throw ( decaf::lang::exceptions::IllegalArgumentException )` [protected]

Creates a **ByteBuffer** (p. 1049) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

### Parameters

<i>capacity</i>	The size of the array, this is the limit we read and write to.
-----------------	--

### Exceptions

<i>IllegalArgumentException</i>	if capacity is negative.
---------------------------------	--------------------------

6.169.2.2 `virtual decaf::nio::ByteBuffer::~~ByteBuffer ( )` [inline, virtual]

## 6.169.3 Member Function Documentation

6.169.3.1 `static ByteBuffer* decaf::nio::ByteBuffer::allocate ( int capacity ) throw ( decaf::lang::exceptions::IllegalArgumentException )` [static]

Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.

### Parameters

<i>capacity</i>	The internal buffer's capacity.
-----------------	---------------------------------

### Returns

a newly allocated **ByteBuffer** (p. 1049) which the caller owns.

### Exceptions

<i>IllegalArgumentException</i>	if capacity is negative.
---------------------------------	--------------------------

6.169.3.2 `virtual unsigned char* decaf::nio::ByteBuffer::array ( ) throw ( ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException )` [pure virtual]

Returns the byte array that backs this buffer.



Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

The array that backs this buffer

### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is backed by an array but is read-only
<b><i>UnsupportedOperationException</i></b>	if this buffer is not backed by an accessible array

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1020).

**6.169.3.3** `virtual int decaf::nio::ByteBuffer::arrayOffset ( )  
const throw ( ReadOnlyBufferException,  
decaf::lang::exceptions::UnsupportedOperationException ) [pure  
virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer.

If this buffer is backed by an array then buffer position `p` corresponds to array index `p + arrayOffset()` (p. 1057).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

The offset within this buffer's array of the first element of the buffer.

### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is backed by an array but is read-only.
<b><i>UnsupportedOperationException</i></b>	if this buffer is not backed by an accessible array.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1020).

**6.169.3.4** `virtual CharBuffer* decaf::nio::ByteBuffer::asCharBuffer ( ) const [pure  
virtual]`

Creates a view of this byte buffer as a char buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the new Char **Buffer** (p. 936), which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 1021).

**6.169.3.5** `virtual DoubleBuffer* decaf::nio::ByteBuffer::asDoubleBuffer ( ) const` [pure virtual]

Creates a view of this byte buffer as a double buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the new double **Buffer** (p. 936), which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 1021).

**6.169.3.6** `virtual FloatBuffer* decaf::nio::ByteBuffer::asFloatBuffer ( ) const` [pure virtual]

Creates a view of this byte buffer as a float buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the new float **Buffer** (p. 936), which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 1021).

**6.169.3.7** `virtual IntBuffer* decaf::nio::ByteBuffer::asIntBuffer ( ) const` `[pure virtual]`

Creates a view of this byte buffer as a int buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the new int **Buffer** (p. 936), which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 1022).

**6.169.3.8** `virtual LongBuffer* decaf::nio::ByteBuffer::asLongBuffer ( ) const` `[pure virtual]`

Creates a view of this byte buffer as a long buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the new long **Buffer** (p. 936), which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 1022).

**6.169.3.9** `virtual ByteBuffer* decaf::nio::ByteBuffer::asReadOnlyBuffer ( ) const` `[pure virtual]`

Creates a new, read-only byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

**Returns**

The new, read-only byte buffer which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1023).

**6.169.3.10** `virtual ShortBuffer* decaf::nio::ByteBuffer::asShortBuffer ( ) const` [pure virtual]

Creates a view of this byte buffer as a short buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns**

the new short **Buffer** (p. 936), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1023).

**6.169.3.11** `virtual ByteBuffer& decaf::nio::ByteBuffer::compact ( ) throw ( ReadOnlyBufferException )` [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 941) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 940) - 1 is copied to index  $n = \text{limit}()$  (p. 940) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

**Returns**

a reference to this **ByteBuffer** (p. 1049).

**Exceptions**

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this buffer is read-only.
--	------------------------------

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1023).

**6.169.3.12** `virtual int decaf::nio::ByteBuffer::compareTo ( const ByteBuffer & value ) const`  
[virtual]

**6.169.3.13** `virtual ByteBuffer* decaf::nio::ByteBuffer::duplicate ( )` [pure virtual]

Creates a new byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

a new Byte **Buffer** (p. 936) which the caller owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1024).

**6.169.3.14** `virtual bool decaf::nio::ByteBuffer::equals ( const ByteBuffer & value ) const`  
[virtual]

**6.169.3.15** `ByteBuffer& decaf::nio::ByteBuffer::get ( std::vector< unsigned char > buffer )`  
`throw ( BufferUnderflowException )`

Relative bulk get method.

This method transfers bytes from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize( N )` before calling this get method.

#### Returns

a reference to this Byte **Buffer** (p. 936).

#### Exceptions

<i><b>BufferUnderflowException</b></i> (p. 967)	if there are fewer than length bytes remaining in this buffer
--	---

**6.169.3.16** `virtual unsigned char decaf::nio::ByteBuffer::get ( ) const throw (`  
`BufferUnderflowException )` [pure virtual]

Relative get method.

Reads the byte at this buffer's current position, and then increments the position.

### Returns

The byte at the buffer's current position.

### Exceptions

<i><b>BufferUnderflowException</b></i> (p. 967)	if the buffer's current position is not smaller than its limit.
--	---

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1024).

**6.169.3.17** `virtual unsigned char decaf::nio::ByteBuffer::get ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Absolute get method.

Reads the byte at the given index.

### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the byte is to be read.
--------------	---

### Returns

the byte that is located at the given index.

### Exceptions

<i><b>IndexOutOfBoundsException</b></i>	if index is not smaller than the buffer's limit, or index is negative.
---	--

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1025).

**6.169.3.18** `ByteBuffer& decaf::nio::ByteBuffer::get ( unsigned char * buffer, int size, int offset, int length ) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )`

Relative bulk get method.

This method transfers bytes from this buffer into the given destination array. If there are fewer bytes remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 941), then no bytes are transferred and a **BufferUnderflowException** (p. 967) is thrown.

Otherwise, this method copies `length` bytes from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of

this buffer is then incremented by length.

#### Parameters

<i>buffer</i>	The pointer to an allocated buffer to fill.
<i>size</i>	The size of the passed in <b>Buffer</b> (p. 936).
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The amount of data to put in the passed buffer.

#### Returns

a reference to this **Buffer** (p. 936).

#### Exceptions

<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.
<b>BufferUnderflowException</b> (p. 967)	if there are fewer than length bytes remaining in this buffer.
<i>NullPointerException</i>	if the passed buffer is null.

**6.169.3.19** virtual char decaf::nio::ByteBuffer::getChar ( ) throw ( **BufferUnderflowException** ) [pure virtual]

Reads the next byte at this buffer's current position, and then increments the position by one.

#### Returns

the next char in the buffer.

#### Exceptions

<b>BufferUnderflowException</b> (p. 967)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 1025).

**6.169.3.20** virtual char decaf::nio::ByteBuffer::getChar ( int *index* ) const throw ( **decaf::lang::exceptions::IndexOutOfBoundsException** ) [pure virtual]

Reads one byte at the given index and returns it.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the byte is to be read.
--------------	---

**Returns**

the char at the given index in the buffer

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1025).

**6.169.3.21** `virtual double decaf::nio::ByteBuffer::getDouble ( ) throw ( BufferUnderflowException )` [pure virtual]

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

**Returns**

the next double in the buffer.

**Exceptions**

<i>BufferUnderflowException</i> (p. 967)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1026).

**6.169.3.22** `virtual double decaf::nio::ByteBuffer::getDouble ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [pure virtual]

Reads eight bytes at the given index and returns it.

**Parameters**

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the bytes are to be read.
--------------	---

**Returns**

the double at the given index in the buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1026).



**6.169.3.23** virtual float decaf::nio::ByteBuffer::getFloat ( ) throw ( **BufferUnderflowException** ) [pure virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

#### Returns

the next float in the buffer.

#### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 967)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
--	---

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1027).

**6.169.3.24** virtual float decaf::nio::ByteBuffer::getFloat ( int *index* ) const throw ( **decaf::lang::exceptions::IndexOutOfBoundsException** ) [pure virtual]

Reads four bytes at the given index and returns it.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the bytes are to be read.
--------------	---

#### Returns

the float at the given index in the buffer.

#### Exceptions

<b><i>IndexOutOfBoundsException</i></b>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
---	--

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1027).

**6.169.3.25** virtual int decaf::nio::ByteBuffer::getInt ( ) throw ( **BufferUnderflowException** ) [pure virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

#### Returns

the next int in the buffer.

**Exceptions**

<i><b>BufferUnderflowException</b></i> (p. 967)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
--	---

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1027).

**6.169.3.26** `virtual int decaf::nio::ByteBuffer::getInt ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [pure virtual]

Reads four bytes at the given index and returns it.

**Parameters**

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the bytes are to be read.
--------------	---

**Returns**

the int at the given index in the buffer.

**Exceptions**

<i><b>IndexOutOfBoundsException</b></i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
---	--

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1028).

**6.169.3.27** `virtual long long decaf::nio::ByteBuffer::getLong ( ) throw ( BufferUnderflowException )` [pure virtual]

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

**Returns**

the next long long in the buffer.

**Exceptions**

<i><b>BufferUnderflowException</b></i> (p. 967)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
--	---

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1028).

**6.169.3.28** `virtual long long decaf::nio::ByteBuffer::getLong ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [pure virtual]

Reads eight bytes at the given index and returns it.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the bytes are to be read.
--------------	---

#### Returns

the long long at the given index in the buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1029).

**6.169.3.29** `virtual short decaf::nio::ByteBuffer::getShort ( ) throw ( BufferUnderflowException )` [pure virtual]

Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

#### Returns

the next short in the buffer.

#### Exceptions

<i>BufferUnderflowException</i> (p. 967)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1029).

**6.169.3.30** `virtual short decaf::nio::ByteBuffer::getShort ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [pure virtual]

Reads two bytes at the given index and returns it.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the bytes are to be read.
--------------	---

**Returns**

the short at the given index in the buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1029).

**6.169.3.31** `virtual bool decaf::nio::ByteBuffer::hasArray ( ) const` [pure virtual]

Tells whether or not this buffer is backed by an accessible byte array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns**

true if, and only if, this buffer is backed by an array and is not read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1030).

**6.169.3.32** `virtual bool decaf::nio::ByteBuffer::isReadOnly ( ) const` [pure virtual]

Tells whether or not this buffer is read-only.

**Returns**

true if, and only if, this buffer is read-only

Implements **decaf::nio::Buffer** (p. 940).

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1030).

**6.169.3.33** `virtual bool decaf::nio::ByteBuffer::operator< ( const ByteBuffer & value ) const` [virtual]

**6.169.3.34** `virtual bool decaf::nio::ByteBuffer::operator== ( const ByteBuffer & value ) const` [virtual]

**6.169.3.35** `virtual ByteBuffer& decaf::nio::ByteBuffer::put ( unsigned char value ) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes the given byte into this buffer at the current position, and then increments the position.

**Parameters**

<i>value</i>	- the byte value to be written.
--------------	---------------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 964)	if this buffer's current position is not smaller than its limit.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1030).

**6.169.3.36** virtual **ByteBuffer&** decaf::nio::ByteBuffer::put ( int *index*, unsigned char *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException ) [pure virtual]

Writes the given byte into this buffer at the given index.

**Parameters**

<i>index</i>	- position in the <b>Buffer</b> (p. 936) to write the data
<i>value</i>	- the byte to write.

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>IndexOutOfBoundsException</i></b>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1031).

**6.169.3.37** **ByteBuffer&** decaf::nio::ByteBuffer::put ( **ByteBuffer & src** ) throw ( **BufferOverflowException**, **ReadOnlyBufferException**, decaf::lang::exceptions::IllegalArgumentException )

This method transfers the bytes remaining in the given source buffer into this buffer.

If there are more bytes remaining in the source buffer than in this buffer, that is, if

`src.remaining() > remaining()` (p. 941), then no bytes are transferred and a **BufferOverflowException** (p. 964) is thrown.

Otherwise, this method copies `n = src.remaining()` bytes from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

#### Parameters

<i>src</i>	The buffer to take bytes from an place in this one.
------------	---

#### Returns

a reference to this buffer

#### Exceptions

<b>BufferOverflowException</b> (p. 964)	if there is insufficient space in this buffer for the remaining bytes in the source buffer
<b>IllegalArgumentException</b>	if the source buffer is this buffer
<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only

**6.169.3.38** `ByteBuffer& decaf::nio::ByteBuffer::put ( const unsigned char * buffer, int size, int offset, int length ) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )`

This method transfers bytes into this buffer from the given source array.

If there are more bytes to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 941), then no bytes are transferred and a **BufferOverflowException** (p. 964) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given `offset` in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

#### Parameters

<i>buffer</i>	The array from which bytes are to be read.
<i>size</i>	The size of the given array.
<i>offset</i>	The offset within the array of the first byte to be read.
<i>length</i>	The number of bytes to be read from the given array.

#### Returns

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 964)	if there is insufficient space in this buffer.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.
<b><i>NullPointerException</i></b>	if the passed buffer is null.
<b><i>IndexOutOfBoundsException</i></b>	if the preconditions of size, offset, or length are not met.

**6.169.3.39 ByteBuffer& decaf::nio::ByteBuffer::put ( std::vector< unsigned char > & *buffer* ) throw ( **BufferOverflowException**, **ReadOnlyBufferException** )**

This method transfers the entire content of the given source byte array into this buffer.

This is the same as calling put( &buffer[0], buffer.size(), 0, buffer.size() )

**Parameters**

<i>buffer</i>	The buffer whose contents are copied to this <b>ByteBuffer</b> (p. 1049).
---------------	---

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 964)	if there is insufficient space in this buffer.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.

**6.169.3.40 virtual ByteBuffer& decaf::nio::ByteBuffer::putChar ( char *value* ) throw ( **BufferOverflowException**, **ReadOnlyBufferException** ) [pure virtual]**

Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

**Parameters**

<i>value</i>	The value to be written.
--------------	--------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 964)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 1032).

**6.169.3.41** `virtual ByteBuffer& decaf::nio::ByteBuffer::putChar ( int index, char value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )` [pure virtual]

Writes one byte containing the given value, into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data.
<i>value</i>	The value to write.

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>IndexOutOfBoundsException</i></b>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 1031).

**6.169.3.42** `virtual ByteBuffer& decaf::nio::ByteBuffer::putDouble ( double value ) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters**

<i>value</i>	The value to be written.
--------------	--------------------------



**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 964)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1033).

**6.169.3.43** `virtual ByteBuffer& decaf::nio::ByteBuffer::putDouble ( int index, double value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )` [pure virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data
<i>value</i>	The value to write.

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>IndexOutOfBoundsException</i></b>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1032).

**6.169.3.44** `virtual ByteBuffer& decaf::nio::ByteBuffer::putFloat ( float value ) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters**

<i>value</i>	The value to be written.
--------------	--------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<i><b>BufferOverflowException</b></i> (p. 964)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 1034).

6.169.3.45 **virtual ByteBuffer& decaf::nio::ByteBuffer::putFloat ( int *index*, float *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )** [pure virtual]

Writes four bytes containing the given value, into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data
<i>value</i>	The value to write.

**Returns**

a reference to this buffer.

**Exceptions**

<i><b>IndexOutOfBoundsException</b></i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 1033).

6.169.3.46 **virtual ByteBuffer& decaf::nio::ByteBuffer::putInt ( int *index*, int *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )** [pure virtual]

Writes four bytes containing the given value, into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data.
<i>value</i>	The value to write.

**Returns**

a reference to this buffer

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1034).

**6.169.3.47** `virtual ByteBuffer& decaf::nio::ByteBuffer::putInt ( int value ) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters**

<i>value</i>	The value to be written.
--------------	--------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b>BufferOverflowException</b> (p. 964)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1035).

**6.169.3.48** `virtual ByteBuffer& decaf::nio::ByteBuffer::putLong ( long long value ) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters**

<i>value</i>	The value to be written.
--------------	--------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<i><b>BufferOverflowException</b></i> (p. 964)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 1035).

**6.169.3.49** virtual **ByteBuffer&** decaf::nio::ByteBuffer::putLong ( int *index*, long long *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException ) [pure virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data.
<i>value</i>	The value to write.

**Returns**

a reference to this buffer.

**Exceptions**

<i><b>IndexOutOfBoundsException</b></i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 1036).

**6.169.3.50** virtual **ByteBuffer&** decaf::nio::ByteBuffer::putShort ( int *index*, short *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException ) [pure virtual]

Writes two bytes containing the given value, into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data
<i>value</i>	The value to write.

**Returns**

a reference to this buffer

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<i>ReadOnlyBufferException</i> (p. 3260)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1037).

**6.169.3.51** `virtual ByteBuffer& decaf::nio::ByteBuffer::putShort ( short value ) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters**

<i>value</i>	The value to be written.
--------------	--------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<i>BufferOverflowException</i> (p. 964)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
<i>ReadOnlyBufferException</i> (p. 3260)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 1036).

**6.169.3.52** `virtual ByteBuffer* decaf::nio::ByteBuffer::slice ( ) const` [pure virtual]

Creates a new byte buffer whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of

bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the newly create **ByteBuffer** (p. 1049) which the caller owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 1038).

**6.169.3.53** `virtual std::string decaf::nio::ByteBuffer::toString ( ) const` [virtual]

### Returns

a std::string describing this object

**6.169.3.54** `static ByteBuffer* decaf::nio::ByteBuffer::wrap ( unsigned char * array, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [static]

Wraps the passed buffer with a new **ByteBuffer** (p. 1049).

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

### Parameters

<i>array</i>	The array that will back the new buffer.
<i>size</i>	The size of the provided array.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

### Returns

a new **ByteBuffer** (p. 1049) that is backed by buffer, caller owns.

### Exceptions

<i>NullPointerException</i>	if the array passed in is NULL.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

**6.169.3.55** `static ByteBuffer* decaf::nio::ByteBuffer::wrap ( std::vector< unsigned char > & buffer ) [static]`

Wraps the passed STL Byte Vector in a **ByteBuffer** (p. 1049).

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling <code>vector.resize( N )</code> .
---------------	--

#### Returns

a new **ByteBuffer** (p. 1049) that is backed by `buffer`, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/ByteBuffer.h`

## 6.170 cms::BytesMessage Class Reference

A **BytesMessage** (p. 1079) object is used to send a message containing a stream of unsigned bytes.

```
#include <src/main/cms/BytesMessage.h>
```

Inheritance diagram for `cms::BytesMessage`:

#### Public Member Functions

- virtual `~BytesMessage ()`
- virtual void **setBodyBytes** (const unsigned char \*buffer, int numBytes)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*sets the bytes given to the message body.*
- virtual unsigned char \* **getBodyBytes** () const =0 throw ( cms::MessageNotReadableException, cms::CMSEException )  
*Gets the bytes that are contained in this message and returns them in a newly allocated array that becomes the property of the caller.*
- virtual int **getBodyLength** () const =0 throw ( cms::MessageNotReadableException, cms::CMSEException )  
*Returns the number of bytes contained in the body of this message.*

- virtual void **reset** ()=0 throw ( cms::MessageFormatException, cms::CMSEException )  
*Puts the message body in read-only mode and repositions the stream of bytes to the beginning.*
- virtual bool **readBoolean** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Boolean from the Bytes message stream.*
- virtual void **writeBoolean** (bool value)=0 throw ( cms::MessageNotWritableException, cms::CMSEException )  
*Writes a boolean to the bytes message stream as a 1-byte value.*
- virtual unsigned char **readByte** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Byte from the Bytes message stream.*
- virtual void **writeByte** (unsigned char value)=0 throw ( cms::MessageNotWritableException, cms::CMSEException )  
*Writes a byte to the bytes message stream as a 1-byte value.*
- virtual int **readBytes** (std::vector< unsigned char > &value) const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a byte array from the bytes message stream.*
- virtual void **writeBytes** (const std::vector< unsigned char > &value)=0 throw ( cms::MessageNotWritableException, cms::CMSEException )  
*Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.*
- virtual int **readBytes** (unsigned char \*buffer, int length) const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a portion of the bytes message stream.*
- virtual void **writeBytes** (const unsigned char \*value, int offset, int length)=0 throw ( cms::MessageNotWritableException, cms::CMSEException )  
*Writes a portion of a byte array to the bytes message stream.*
- virtual char **readChar** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Char from the Bytes message stream.*
- virtual void **writeChar** (char value)=0 throw ( cms::MessageNotWritableException, cms::CMSEException )  
*Writes a char to the bytes message stream as a 1-byte value.*



- virtual float **readFloat** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 32 bit float from the Bytes message stream.*
- virtual void **writeFloat** (float value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a float to the bytes message stream as a 4 byte value.*
- virtual double **readDouble** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 64 bit double from the Bytes message stream.*
- virtual void **writeDouble** (double value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a double to the bytes message stream as a 8 byte value.*
- virtual short **readShort** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 16 bit signed short from the Bytes message stream.*
- virtual void **writeShort** (short value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a signed short to the bytes message stream as a 2 byte value.*
- virtual unsigned short **readUnsignedShort** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 16 bit unsigned short from the Bytes message stream.*
- virtual void **writeUnsignedShort** (unsigned short value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a unsigned short to the bytes message stream as a 2 byte value.*
- virtual int **readInt** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 32 bit signed integer from the Bytes message stream.*
- virtual void **writeInt** (int value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a signed int to the bytes message stream as a 4 byte value.*
- virtual long long **readLong** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 64 bit long from the Bytes message stream.*
- virtual void **writeLong** (long long value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a long long to the bytes message stream as a 8 byte value.*

- virtual std::string **readString** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads an ASCII String from the Bytes message stream.*

- virtual void **writeString** (const std::string &value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes an ASCII String to the Bytes message stream.*

- virtual std::string **readUTF** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads an UTF String from the **BytesMessage** (p. 1079) stream.*

- virtual void **writeUTF** (const std::string &value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes an UTF String to the **BytesMessage** (p. 1079) stream.*

- virtual **BytesMessage** \* **clone** () const =0

*Clones this message.*

### 6.170.1 Detailed Description

A **BytesMessage** (p. 1079) object is used to send a message containing a stream of unsigned bytes. It inherits from the **Message** (p. 2614) interface and adds a bytes message body. The receiver of the message supplies the interpretation of the bytes using the methods added by the **BytesMessage** (p. 1079) interface.

The **BytesMessage** (p. 1079) methods are based largely on those found in **decaf.io.DataInputStream** (p. 1612) and **decaf.io.DataOutputStream** (p. 1628).

Although the CMS API allows the use of message properties with byte messages, they are typically not used, since the inclusion of properties may affect the format.

The primitive types can be written explicitly using methods for each type. Because the C++ language is more limited when dealing with primitive types the JMS equivalent generic read and write methods that take Java objects cannot be provided in the CMS API.

When the message is first created, and when **clearBody** is called, the body of the message is in write-only mode. After the first call to **reset** has been made, the message body is in read-only mode. After a message has been sent, the client that sent it can retain and modify it without affecting the message that has been sent. The same message object can be sent multiple times. When a message has been received, the provider has called **reset** so that the message body is in read-only mode for the client.

If **clearBody** is called on a message in read-only mode, the message body is cleared and the message is in write-only mode.

If a client attempts to read a message in write-only mode, a **MessageNotReadableException** (p. 2807) is thrown.

If a client attempts to write a message in read-only mode, a **MessageNotWriteableException** (p. 2808) is thrown.

#### Since

1.0

### 6.170.2 Constructor & Destructor Documentation

6.170.2.1 `virtual cms::ByteMessage::~ByteMessage ( ) [inline, virtual]`

### 6.170.3 Member Function Documentation

6.170.3.1 `virtual ByteMessage* cms::ByteMessage::clone ( ) const [pure virtual]`

Clones this message.

#### Returns

a deep copy of this message.

#### Exceptions

<i><b>CMSEException</b></i> (p. 1190)	- if an internal error occurs while cloning the <b>Message</b> (p. 2614).
--	---

Implements **cms::Message** (p. 2620).

Implemented in **activemq::commands::ActiveMQByteMessage** (p. 219).

6.170.3.2 `virtual unsigned char* cms::ByteMessage::getBodyBytes ( ) const throw ( cms::MessageNotReadableException, cms::CMSEException ) [pure virtual]`

Gets the bytes that are contained in this message and returns them in a newly allocated array that becomes the property of the caller.

This is a copy of the data contained in this message, changing the value contained in this array has no effect on the data contained in this message.

#### Returns

pointer to a byte buffer that the call owns upon completion of this method.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- If an internal error occurs.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- If the message is in Write Only Mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 220).

**6.170.3.3** `virtual int cms::BytesMessage::getBodyLength ( ) const throw ( cms::MessageNotReadableException, cms::CMSEException )` [pure virtual]

Returns the number of bytes contained in the body of this message.

#### Returns

number of bytes.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- If an internal error occurs.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- If the message is in Write Only Mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 221).

**6.170.3.4** `virtual bool cms::BytesMessage::readBoolean ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )` [pure virtual]

Reads a Boolean from the Bytes message stream.

#### Returns

boolean value from stream

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 221).

**6.170.3.5** `virtual unsigned char cms::BytesMessage::readByte ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [pure virtual]`

Reads a Byte from the Bytes message stream.

#### Returns

unsigned char value from stream

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 222).

**6.170.3.6** `virtual int cms::BytesMessage::readBytes ( std::vector< unsigned char > & value ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [pure virtual]`

Reads a byte array from the bytes message stream.

If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

#### Parameters

<i>value</i>	buffer to place data in
--------------	-------------------------

#### Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 222).

```
6.170.3.7 virtual int cms::BytesMessage::readBytes ( unsigned char *
            buffer, int length ) const throw ( cms::MessageEOFException,
            cms::MessageNotReadableException, cms::CMSException ) [pure
            virtual]
```

Reads a portion of the bytes message stream.

If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an `IndexOutOfBoundsException` is thrown. No bytes will be read from the stream for this exception case.

**Parameters**

<i>buffer</i>	the buffer into which the data is read
<i>length</i>	the number of bytes to read; must be less than or equal to value.length

**Returns**

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 223).

**6.170.3.8** `virtual char cms::BytesMessage::readChar ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads a Char from the Bytes message stream.

#### Returns

char value from stream

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 224).

**6.170.3.9** `virtual double cms::BytesMessage::readDouble ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads a 64 bit double from the Bytes message stream.

#### Returns

double value from stream

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 224).

**6.170.3.10** `virtual float cms::BytesMessage::readFloat ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [pure virtual]`

Reads a 32 bit float from the Bytes message stream.

#### Returns

double value from stream

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 225).

**6.170.3.11** `virtual int cms::BytesMessage::readInt ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [pure virtual]`

Reads a 32 bit signed integer from the Bytes message stream.

#### Returns

int value from stream

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 225).



**6.170.3.12** virtual long long cms::BytesMessage::readLong ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [pure virtual]

Reads a 64 bit long from the Bytes message stream.

#### Returns

long long value from stream

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 226).

**6.170.3.13** virtual short cms::BytesMessage::readShort ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [pure virtual]

Reads a 16 bit signed short from the Bytes message stream.

#### Returns

short value from stream

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 226).

**6.170.3.14** `virtual std::string cms::BytesMessage::readString ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [pure virtual]`

Reads an ASCII String from the Bytes message stream.

#### Returns

String from stream

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 227).

**6.170.3.15** `virtual unsigned short cms::BytesMessage::readUnsignedShort ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [pure virtual]`

Reads a 16 bit unsigned short from the Bytes message stream.

#### Returns

unsigned short value from stream

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 227).

**6.170.3.16** virtual std::string cms::BytesMessage::readUTF ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]

Reads an UTF String from the **BytesMessage** (p. 1079) stream.

#### Returns

String from stream

#### Exceptions

<b>CMSException</b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b>MessageEOFException</b> (p. 2747)	- if unexpected end of bytes stream has been reached.
<b>MessageNotReadableException</b> (p. 2807)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 228).

**6.170.3.17** virtual void cms::BytesMessage::reset ( ) throw ( cms::MessageFormatException, cms::CMSException ) [pure virtual]

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

#### Exceptions

<b>CMSException</b> (p. 1190)	- If the provider fails to perform the reset operation.
<b>MessageFormatException</b> (p. 2749)	- If the <b>Message</b> (p. 2614) has an invalid format.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 228).

**6.170.3.18** virtual void cms::BytesMessage::setBodyBytes ( const unsigned char \* buffer, int numBytes ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]

sets the bytes given to the message body.

#### Parameters

<i>buffer</i>	Byte Buffer to copy
<i>numBytes</i>	Number of bytes in Buffer to copy

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- If an internal error occurs.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if in Read Only Mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 228).

**6.170.3.19** `virtual void cms::BytesMessage::writeBoolean ( bool value ) throw ( cms::MessageNotWriteableException, cms::CMSEException )` [pure virtual]

Writes a boolean to the bytes message stream as a 1-byte value.

The value true is written as the value (byte)1; the value false is written as the value (byte)0.

**Parameters**

<i>value</i>	boolean to write to the stream
--------------	--------------------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 229).

**6.170.3.20** `virtual void cms::BytesMessage::writeByte ( unsigned char value ) throw ( cms::MessageNotWriteableException, cms::CMSEException )` [pure virtual]

Writes a byte to the bytes message stream as a 1-byte value.

**Parameters**

<i>value</i>	byte to write to the stream
--------------	-----------------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 229).

**6.170.3.21** `virtual void cms::BytesMessage::writeBytes ( const std::vector< unsigned char > & value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

#### Parameters

<i>value</i>	bytes to write to the stream
--------------	------------------------------

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 230).

**6.170.3.22** `virtual void cms::BytesMessage::writeBytes ( const unsigned char * value, int offset, int length ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a portion of a byte array to the bytes message stream.  
size as the number of bytes to write.

#### Parameters

<i>value</i>	bytes to write to the stream
<i>offset</i>	the initial offset within the byte array
<i>length</i>	the number of bytes to use

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 230).

**6.170.3.23** `virtual void cms::BytesMessage::writeChar ( char value ) throw ( cms::MessageNotWriteableException, cms::CMSEException )` [pure virtual]

Writes a char to the bytes message stream as a 1-byte value.

#### Parameters

<i>value</i>	char to write to the stream
--------------	-----------------------------

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 231).

**6.170.3.24** `virtual void cms::BytesMessage::writeDouble ( double value ) throw ( cms::MessageNotWriteableException, cms::CMSEException )` [pure virtual]

Writes a double to the bytes message stream as a 8 byte value.

#### Parameters

<i>value</i>	double to write to the stream
--------------	-------------------------------

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 231).

**6.170.3.25** `virtual void cms::BytesMessage::writeFloat ( float value ) throw ( cms::MessageNotWriteableException, cms::CMSEException )` [pure virtual]

Writes a float to the bytes message stream as a 4 byte value.

#### Parameters

<i>value</i>	float to write to the stream
--------------	------------------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 232).

**6.170.3.26** `virtual void cms::BytesMessage::writeInt ( int value ) throw ( cms::MessageNotWriteableException, cms::CMSEException )` [pure virtual]

Writes a signed int to the bytes message stream as a 4 byte value.

**Parameters**

<i>value</i>	signed int to write to the stream
--------------	-----------------------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 232).

**6.170.3.27** `virtual void cms::BytesMessage::writeLong ( long long value ) throw ( cms::MessageNotWriteableException, cms::CMSEException )` [pure virtual]

Writes a long long to the bytes message stream as a 8 byte value.

**Parameters**

<i>value</i>	signed long long to write to the stream
--------------	---

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 232).

**6.170.3.28** `virtual void cms::BytesMessage::writeShort ( short value ) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Writes a signed short to the bytes message stream as a 2 byte value.

#### Parameters

<i>value</i>	signed short to write to the stream
--------------	-------------------------------------

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 233).

**6.170.3.29** `virtual void cms::BytesMessage::writeString ( const std::string & value ) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Writes an ASCII String to the Bytes message stream.

#### Parameters

<i>value</i>	String to write to the stream
--------------	-------------------------------

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 233).

**6.170.3.30** `virtual void cms::BytesMessage::writeUnsignedShort ( unsigned short value ) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Writes a unsigned short to the bytes message stream as a 2 byte value.

#### Parameters

<i>value</i>	unsigned short to write to the stream
--------------	---------------------------------------



**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWritableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 234).

**6.170.3.31** `virtual void cms::BytesMessage::writeUTF ( const std::string & value ) throw ( cms::MessageNotWritableException, cms::CMSEException ) [pure virtual]`

Writes an UTF String to the **BytesMessage** (p. 1079) stream.

**Parameters**

<i>value</i>	String to write to the stream
--------------	-------------------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 234).

The documentation for this class was generated from the following file:

- src/main/cms/BytesMessage.h

**6.171 activemq::cmsutil::CachedConsumer Class Reference**

A cached message consumer contained within a pooled session.

```
#include <src/main/activemq/cmsutil/CachedConsumer.h>
```

Inheritance diagram for **activemq::cmsutil::CachedConsumer**:

**Public Member Functions**

- **CachedConsumer** (cms::MessageConsumer \*consumer)

- virtual `~CachedConsumer ()`
- virtual void `close ()` throw ( cms::CMSEException )  
*Does nothing - the real producer resource will be closed by the lifecycle manager.*
- virtual `cms::Message * receive ()` throw ( cms::CMSEException )  
*Synchronously Receive a Message.*
- virtual `cms::Message * receive (int millisecs)` throw ( cms::CMSEException )  
*Synchronously Receive a Message, time out after defined interval.*
- virtual `cms::Message * receiveNoWait ()` throw ( cms::CMSEException )  
*Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.*
- virtual void `setMessageListener (cms::MessageListener *listener)` throw ( cms::CMSEException )  
*Sets the MessageListener that this class will send notifis on.*
- virtual `cms::MessageListener * getMessageListener ()` const throw ( cms::CMSEException )  
*Gets the MessageListener that this class will send mew Message notification events to.*
- virtual std::string `getMessageSelector ()` const throw ( cms::CMSEException )  
*Gets this message consumer's message selector expression.*

### Protected Member Functions

- `CachedConsumer (const CachedConsumer &)`
- `CachedConsumer & operator= (const CachedConsumer &)`

#### 6.171.1 Detailed Description

A cached message consumer contained within a pooled session.

### 6.171.2 Constructor & Destructor Documentation

6.171.2.1 `activemq::cmsutil::CachedConsumer::CachedConsumer ( const CachedConsumer & ) [inline, protected]`

6.171.2.2 `activemq::cmsutil::CachedConsumer::CachedConsumer ( cms::MessageConsumer * consumer ) [inline]`

6.171.2.3 `virtual activemq::cmsutil::CachedConsumer::~~CachedConsumer ( ) [inline, virtual]`

### 6.171.3 Member Function Documentation

6.171.3.1 `virtual void activemq::cmsutil::CachedConsumer::close ( ) throw ( cms::CMSEException ) [inline, virtual]`

Does nothing - the real producer resource will be closed by the lifecycle manager.

Implements `cms::Closeable` (p. 1180).

6.171.3.2 `virtual cms::MessageListener* activemq::cmsutil::CachedConsumer::getMessageListener ( ) const throw ( cms::CMSEException ) [inline, virtual]`

Gets the MessageListener that this class will send new Message notification events to.

#### Returns

The listener of messages received by this consumer

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements `cms::MessageConsumer` (p. 2675).

6.171.3.3 `virtual std::string activemq::cmsutil::CachedConsumer::getMessageSelector ( ) const throw ( cms::CMSEException ) [inline, virtual]`

Gets this message consumer's message selector expression.

#### Returns

This Consumer's selector expression or "".

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageConsumer** (p. 2676).

**6.171.3.4** **CachedConsumer& activemq::cmsutil::CachedConsumer::operator= ( const  
CachedConsumer & )** [inline, protected]

**6.171.3.5** **virtual cms::Message\* activemq::cmsutil::CachedConsumer::receive ( ) throw ( cms::CMSEException )** [inline, virtual]

Synchronously Receive a Message.

#### Returns

new message which the caller owns and must delete.

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageConsumer** (p. 2676).

**6.171.3.6** **virtual cms::Message\* activemq::cmsutil::CachedConsumer::receive ( int millisecs  
) throw ( cms::CMSEException )** [inline, virtual]

Synchronously Receive a Message, time out after defined interval.

Returns null if nothing read.

#### Returns

new message which the caller owns and must delete.

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageConsumer** (p. 2676).

**6.171.3.7** **virtual cms::Message\* activemq::cmsutil::CachedConsumer::receiveNoWait ( )  
throw ( cms::CMSEException )** [inline, virtual]

Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.

#### Returns

new message which the caller owns and must delete.

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageConsumer** (p. 2677).

**6.171.3.8** `virtual void activemq::cmsutil::CachedConsumer::setMessageListener ( cms::MessageListener * listener ) throw ( cms::CMSEException )`  
`[inline, virtual]`

Sets the MessageListener that this class will send notifs on.

#### Parameters

<i>listener</i>	The listener of messages received by this consumer.
-----------------	---

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageConsumer** (p. 2677).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/CachedConsumer.h

## 6.172 activemq::cmsutil::CachedProducer Class Reference

A cached message producer contained within a pooled session.

```
#include <src/main/activemq/cmsutil/CachedProducer.h>
```

Inheritance diagram for activemq::cmsutil::CachedProducer:

#### Public Member Functions

- **CachedProducer** (**cms::MessageProducer** \*producer)
- virtual **~CachedProducer** ()
- virtual void **close** () throw ( cms::CMSEException )  
*Does nothing - the real producer resource will be closed by the lifecycle manager.*
- virtual void **send** (**cms::Message** \*message) throw ( cms::CMSEException )  
*Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **send** (**cms::Message** \*message, int deliveryMode, int priority, long long timeToLive) throw ( cms::CMSEException )  
*Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.*

- virtual void **send** (const **cms::Destination** \*destination, **cms::Message** \*message) throw ( cms::CMSEException )  
*Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **send** (const **cms::Destination** \*destination, **cms::Message** \*message, int deliveryMode, int priority, long long timeToLive) throw ( cms::CMSEException )  
*Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **setDeliveryMode** (int mode) throw ( cms::CMSEException )  
*Sets the delivery mode for this Producer.*
- virtual int **getDeliveryMode** () const throw ( cms::CMSEException )  
*Gets the delivery mode for this Producer.*
- virtual void **setDisableMessageID** (bool value) throw ( cms::CMSEException )  
*Sets if Message Ids are disabled for this Producer.*
- virtual bool **getDisableMessageID** () const throw ( cms::CMSEException )  
*Gets if Message Ids are disabled for this Producer.*
- virtual void **setDisableMessageTimeStamp** (bool value) throw ( cms::CMSEException )  
*Sets if Message Time Stamps are disabled for this Producer.*
- virtual bool **getDisableMessageTimeStamp** () const throw ( cms::CMSEException )  
*Gets if Message Time Stamps are disabled for this Producer.*
- virtual void **setPriority** (int priority) throw ( cms::CMSEException )  
*Sets the Priority that this Producers sends messages at.*
- virtual int **getPriority** () const throw ( cms::CMSEException )  
*Gets the Priority level that this producer sends messages at.*
- virtual void **setTimeToLive** (long long time) throw ( cms::CMSEException )  
*Sets the Time to Live that this Producers sends messages with.*
- virtual long long **getTimeToLive** () const throw ( cms::CMSEException )  
*Gets the Time to Live that this producer sends messages with.*

### Protected Member Functions

- **CachedProducer** (const **CachedProducer** &)
- **CachedProducer** & **operator=** (const **CachedProducer** &)

### 6.172.1 Detailed Description

A cached message producer contained within a pooled session.

### 6.172.2 Constructor & Destructor Documentation

**6.172.2.1** `activemq::cmsutil::CachedProducer::CachedProducer ( const CachedProducer & ) [inline, protected]`

**6.172.2.2** `activemq::cmsutil::CachedProducer::CachedProducer ( cms::MessageProducer * producer ) [inline]`

**6.172.2.3** `virtual activemq::cmsutil::CachedProducer::~~CachedProducer ( ) [inline, virtual]`

### 6.172.3 Member Function Documentation

**6.172.3.1** `virtual void activemq::cmsutil::CachedProducer::close ( ) throw ( cms::CMSEException ) [inline, virtual]`

Does nothing - the real producer resource will be closed by the lifecycle manager.

Implements `cms::Closeable` (p. 1180).

**6.172.3.2** `virtual int activemq::cmsutil::CachedProducer::getDeliveryMode ( ) const throw ( cms::CMSEException ) [inline, virtual]`

Gets the delivery mode for this Producer.

#### Returns

The DeliveryMode

#### Exceptions

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements `cms::MessageProducer` (p. 2811).

**6.172.3.3** `virtual bool activemq::cmsutil::CachedProducer::getDisableMessageID ( ) const throw ( cms::CMSEException ) [inline, virtual]`

Gets if Message Ids are disabled for this Producer.

#### Returns

boolean indicating enable / disable (true / false)

**Exceptions**

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 2812).

**6.172.3.4** `virtual bool activemq::cmsutil::CachedProducer::getDisableMessageTimeStamp ( )  
const throw ( cms::CMSEException ) [inline, virtual]`

Gets if Message Time Stamps are disabled for this Producer.

**Returns**

boolean indicating enable / disable (true / false)

**Exceptions**

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 2812).

**6.172.3.5** `virtual int activemq::cmsutil::CachedProducer::getPriority ( ) const throw ( cms::CMSEException ) [inline, virtual]`

Gets the Priority level that this producer sends messages at.

**Returns**

int based priority level

**Exceptions**

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 2812).

**6.172.3.6** `virtual long long activemq::cmsutil::CachedProducer::getTimeToLive ( ) const throw ( cms::CMSEException ) [inline, virtual]`

Gets the Time to Live that this producer sends messages with.

**Returns**

Time to live value in milliseconds

**Exceptions**

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------



Implements **cms::MessageProducer** (p. 2813).

**6.172.3.7** **CachedProducer& activemq::cmsutil::CachedProducer::operator= ( const CachedProducer & )** [inline, protected]

**6.172.3.8** **virtual void activemq::cmsutil::CachedProducer::send ( cms::Message \* message, int deliveryMode, int priority, long long timeToLive ) throw ( cms::CMSEException )** [inline, virtual]

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

#### Parameters

<i>message</i>	The message to be sent.
<i>delivery-Mode</i>	The delivery mode to be used.
<i>priority</i>	The priority for this message.
<i>timeToLive</i>	The time to live value for this message in milliseconds.

#### Exceptions

<i>CMSEException</i>	- if an internal error occurs while sending the message.
<i>MessageFormatException</i>	- if an Invalid Message is given.
<i>InvalidDestinationException</i>	- if a client uses this method with a MessageProducer with an invalid destination.
<i>UnsupportedOperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2813).

**6.172.3.9** **virtual void activemq::cmsutil::CachedProducer::send ( const cms::Destination \* destination, cms::Message \* message ) throw ( cms::CMSEException )** [inline, virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

#### Parameters

<i>destination</i>	The destination on which to send the message
<i>message</i>	the message to be sent.

#### Exceptions

<i>CMSEException</i>	- if an internal error occurs while sending the message.
----------------------	--

<i>MessageFormatException</i>	- if an Invalid Message is given.
<i>InvalidDestinationException</i>	- if a client uses this method with a MessageProducer with an invalid destination.
<i>UnsupportedOperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2815).

**6.172.3.10** `virtual void activemq::cmsutil::CachedProducer::send ( cms::Message * message ) throw ( cms::CMSException ) [inline, virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

#### Parameters

<i>message</i>	The message to be sent.
----------------	-------------------------

#### Exceptions

<i>CMSException</i>	- if an internal error occurs while sending the message.
<i>MessageFormatException</i>	- if an Invalid Message is given.
<i>InvalidDestinationException</i>	- if a client uses this method with a MessageProducer with an invalid destination.
<i>UnsupportedOperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2814).

**6.172.3.11** `virtual void activemq::cmsutil::CachedProducer::send ( const cms::Destination * destination, cms::Message * message, int deliveryMode, int priority, long long timeToLive ) throw ( cms::CMSException ) [inline, virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

#### Parameters

<i>destination</i>	The destination on which to send the message
<i>message</i>	The message to be sent.
<i>delivery-Mode</i>	The delivery mode to be used.
<i>priority</i>	The priority for this message.
<i>timeToLive</i>	The time to live value for this message in milliseconds.

**Exceptions**

<i>CMSEException</i>	- if an internal error occurs while sending the message.
<i>MessageFormatException</i>	- if an Invalid Message is given.
<i>InvalidDestinationException</i>	- if a client uses this method with a MessageProducer with an invalid destination.
<i>UnsupportedOperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2814).

**6.172.3.12** `virtual void activemq::cmsutil::CachedProducer::setDeliveryMode ( int mode ) throw ( cms::CMSEException ) [inline, virtual]`

Sets the delivery mode for this Producer.

**Parameters**

<i>mode</i>	The DeliveryMode
-------------	------------------

**Exceptions**

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 2816).

**6.172.3.13** `virtual void activemq::cmsutil::CachedProducer::setDisableMessageID ( bool value ) throw ( cms::CMSEException ) [inline, virtual]`

Sets if Message Ids are disabled for this Producer.

**Parameters**

<i>value</i>	boolean indicating enable / disable (true / false)
--------------	--

**Exceptions**

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 2816).

**6.172.3.14** `virtual void activemq::cmsutil::CachedProducer::setDisableMessageTimeStamp ( bool value ) throw ( cms::CMSEException ) [inline, virtual]`

Sets if Message Time Stamps are disabled for this Producer.

**Parameters**

<i>value</i>	- boolean indicating enable / disable (true / false)
--------------	--

**Exceptions**

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 2816).

**6.172.3.15** `virtual void activemq::cmsutil::CachedProducer::setPriority ( int priority ) throw ( cms::CMSEException )` [inline, virtual]

Sets the Priority that this Producers sends messages at.

**Parameters**

<i>priority</i>	int value for Priority level
-----------------	------------------------------

**Exceptions**

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 2817).

**6.172.3.16** `virtual void activemq::cmsutil::CachedProducer::setTimeToLive ( long long time ) throw ( cms::CMSEException )` [inline, virtual]

Sets the Time to Live that this Producers sends messages with.

This value will be used if the time to live is not specified via the send method.

**Parameters**

<i>time</i>	default time to live value in milliseconds
-------------	--

**Exceptions**

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 2817).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CachedProducer.h`

## 6.173 decaf::util::concurrent::Callable< V > Class Template Reference

A task that returns a result and may throw an exception.

```
#include <src/main/decaf/util/concurrent/Callable.h>
```

## Public Member Functions

- virtual `~Callable()`
- virtual `V call()`=0 throw ( decaf::lang::Exception )

*Computes a result, or throws an exception if unable to do so.*

### 6.173.1 Detailed Description

`template<typename V> class decaf::util::concurrent::Callable< V >`

A task that returns a result and may throw an exception. Implementors define a single method with no arguments called `call`. This interface differs from the `Runnable` interface in that a **Callable** (p. 1108) object can return a result and is allowed to throw an exceptions from its `call` method.

The `Executors` class contains utility methods to convert from other common forms to **Callable** (p. 1108) classes.

#### Since

1.0

### 6.173.2 Constructor & Destructor Documentation

6.173.2.1 `template<typename V> virtual decaf::util::concurrent::Callable< V >::~~Callable( )` [inline, virtual]

### 6.173.3 Member Function Documentation

6.173.3.1 `template<typename V> virtual V decaf::util::concurrent::Callable< V >::call( )` throw ( decaf::lang::Exception ) [pure virtual]

Computes a result, or throws an exception if unable to do so.

#### Returns

Computed Result.

#### Exceptions

<i>Exception</i>	If unable to compute a result.
------------------	--------------------------------

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Callable.h`

## 6.174 decaf::util::concurrent::CancellationException Class Reference

```
#include <src/main/decaf/util/concurrent/CancellationException.h>
```

Inheritance diagram for decaf::util::concurrent::CancellationException:

### Public Member Functions

- **CancellationException** () throw ()  
*Default Constructor.*
- **CancellationException** (const decaf::lang::Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **CancellationException** (const CancellationException &ex) throw ()  
*Copy Constructor.*
- **CancellationException** (const std::exception \*cause) throw ()  
*Constructor.*
- **CancellationException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **CancellationException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **CancellationException** \* clone () const  
*Clones this exception.*
- virtual ~**CancellationException** () throw ()

### 6.174.1 Constructor & Destructor Documentation

6.174.1.1 decaf::util::concurrent::CancellationException::CancellationException ( ) throw ()  
[inline]

Default Constructor.

6.174.1.2 decaf::util::concurrent::CancellationException::CancellationException ( const decaf::lang::Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

**Parameters**

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.174.1.3** `decaf::util::concurrent::CancellationException::CancellationException ( const CancellationException & ex ) throw () [inline]`

Copy Constructor.

**Parameters**

<i>ex</i>	- The Exception to copy in this new instance.
-----------	---

**6.174.1.4** `decaf::util::concurrent::CancellationException::CancellationException ( const std::exception * cause ) throw () [inline]`

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.174.1.5** `decaf::util::concurrent::CancellationException::CancellationException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.
<i>msg</i>	- The message to report
<i>...</i>	- list of primitives that are formatted into the message

**6.174.1.6** `decaf::util::concurrent::CancellationException::CancellationException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	- The file name where exception occurs
-------------	--

<i>lineNumber</i>	- The line number where the exception occurred.
<i>cause</i>	- The exception that was the cause for this one to be thrown.
<i>msg</i>	- The message to report
...	- list of primitives that are formatted into the message

6.174.1.7 `virtual decaf::util::concurrent::CancellationException::~~CancellationException ( )  
throw () [inline, virtual]`

## 6.174.2 Member Function Documentation

6.174.2.1 `virtual CancellationException* de-  
caf::util::concurrent::CancellationException::clone ( ) const  
[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

a new instance of an exception that is a clone of this one.

Reimplemented from `decaf::lang::Exception` (p. 1889).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/CancellationException.h`

## 6.175 decaf::security::cert::Certificate Class Reference

Base interface for all identity certificates.

```
#include <src/main/decaf/security/cert/Certificate.h>
```

Inheritance diagram for `decaf::security::cert::Certificate`:

### Public Member Functions

- `virtual ~Certificate ()`
- `virtual bool equals (const Certificate &cert) const =0`  
*Compares the encoded form of the two certificates.*
- `virtual void getEncoded (std::vector< unsigned char > &output) const =0 throw  
( CertificateEncodingException )`



*Provides the encoded form of this certificate.*

- virtual std::string **getType** () const =0  
*Returns the type of this certificate.*
- virtual **PublicKey** \* **getPublicKey** ()=0  
*Gets the public key of this certificate.*
- virtual const **PublicKey** \* **getPublicKey** () const =0  
*Gets the public key of this certificate.*
- virtual void **verify** (const **PublicKey** &publicKey) const =0 throw ( NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException)  
*Verifies that this certificate was signed with the private key that corresponds to the specified public key.*
- virtual void **verify** (const **PublicKey** &publicKey, const std::string &sigProvider) const =0 throw ( NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException)  
*Verifies that this certificate was signed with the private key that corresponds to the specified public key.*
- virtual std::string **toString** () const =0  
*Returns a string representation of this certificate.*

### 6.175.1 Detailed Description

Base interface for all identity certificates.

### 6.175.2 Constructor & Destructor Documentation

6.175.2.1 virtual decaf::security::cert::Certificate::~Certificate ( ) [inline, virtual]

### 6.175.3 Member Function Documentation

6.175.3.1 virtual bool decaf::security::cert::Certificate::equals ( const Certificate & cert ) const [pure virtual]

Compares the encoded form of the two certificates.

#### Parameters

<i>cert</i>	The certificate to be tested for equality with this certificate.
-------------	--

**Returns**

true if the given certificate is equal to this certificate.

**6.175.3.2** `virtual void decaf::security::cert::Certificate::getEncoded ( std::vector< unsigned char > & output ) const throw ( CertificateEncodingException )` [pure virtual]

Provides the encoded form of this certificate.

**Parameters**

<i>output</i>	Receives the encoded form of this certificate.
---------------	--

**Exceptions**

<b><i>CertificateEncodingException</i></b> (p. 1116)	if an encoding error occurs
---	-----------------------------

**6.175.3.3** `virtual PublicKey* decaf::security::cert::Certificate::getPublicKey ( )` [pure virtual]

Gets the public key of this certificate.

**Returns**

the public key

**6.175.3.4** `virtual const PublicKey* decaf::security::cert::Certificate::getPublicKey ( ) const` [pure virtual]

Gets the public key of this certificate.

**Returns**

the public key

**6.175.3.5** `virtual std::string decaf::security::cert::Certificate::getType ( ) const` [pure virtual]

Returns the type of this certificate.

**Returns**

the type of this certificate

**6.175.3.6** `virtual std::string decaf::security::cert::Certificate::toString ( ) const [pure virtual]`

Returns a string representation of this certificate.

#### Returns

a string representation of this certificate

**6.175.3.7** `virtual void decaf::security::cert::Certificate::verify ( const PublicKey & publicKey, const std::string & sigProvider ) const throw ( NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException) [pure virtual]`

Verifies that this certificate was signed with the private key that corresponds to the specified public key.

Uses the verification engine of the specified provider.

#### Parameters

<i>publicKey</i>	The public key used to carry out the validation.
<i>sigProvider</i>	The name of the signature provider

#### Exceptions

<b><i>NoSuchAlgorithmException</i></b> (p. 2907)	- on unsupported signature algorithms.
<b><i>InvalidKeyException</i></b> (p. 2201)	- on incorrect key.
<b><i>NoSuchProviderException</i></b> (p. 2913)	- if there's no default provider.
<b><i>SignatureException</i></b> (p. 3601)	- on signature errors.
<b><i>CertificateException</i></b> (p. 1118)	- on encoding errors.

**6.175.3.8** `virtual void decaf::security::cert::Certificate::verify ( const PublicKey & publicKey ) const throw ( NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException) [pure virtual]`

Verifies that this certificate was signed with the private key that corresponds to the specified public key.

**Parameters**

<i>publicKey</i>	The public key used to carry out the validation.
------------------	--

**Exceptions**

<b><i>NoSuchAlgorithmException</i></b> (p. 2907)	- on unsupported signature algorithms.
<b><i>InvalidKeyException</i></b> (p. 2201)	- on incorrect key.
<b><i>NoSuchProviderException</i></b> (p. 2913)	- if there's no default provider.
<b><i>SignatureException</i></b> (p. 3601)	- on signature errors.
<b><i>CertificateException</i></b> (p. 1118)	- on encoding errors.

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**Certificate.h**

## 6.176 decaf::security::cert::CertificateEncodingException Class Reference

```
#include <src/main/decaf/security/cert/CertificateEncodingException.h>
```

Inheritance diagram for decaf::security::cert::CertificateEncodingException:

**Public Member Functions**

- **CertificateEncodingException** () throw ()  
*Default Constructor.*
- **CertificateEncodingException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **CertificateEncodingException** (const **CertificateEncodingException** &ex) throw ()  
*Copy Constructor.*
- **CertificateEncodingException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*

- virtual **CertificateEncodingException** \* **clone** () const  
*Clones this exception.*
- virtual ~**CertificateEncodingException** () throw ()

### 6.176.1 Constructor & Destructor Documentation

6.176.1.1 **decaf::security::cert::CertificateEncodingException::CertificateEncodingException** (  
) throw () [inline]

Default Constructor.

6.176.1.2 **decaf::security::cert::CertificateEncodingException::CertificateEncodingException** (  
const Exception & *ex* ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.176.1.3 **decaf::security::cert::CertificateEncodingException::CertificateEncodingException** (  
const **CertificateEncodingException** & *ex* ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.176.1.4 **decaf::security::cert::CertificateEncodingException::CertificateEncodingException**  
( const char \* *file*, const int *lineNumber*, const char \* *msg*, ... ) throw ()  
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
...	list of primitives that are formatted into the message

6.176.1.5 `virtual decaf::security::cert::CertificateEncodingException::~~CertificateEncodingException ( ) throw () [inline, virtual]`

## 6.176.2 Member Function Documentation

6.176.2.1 `virtual CertificateEncodingException* decaf::security::cert::CertificateEncodingException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

A deep copy of this exception.

Reimplemented from `decaf::security::cert::CertificateException` (p. 1120).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateEncodingException.h`

## 6.177 decaf::security::cert::CertificateException Class Reference

```
#include <src/main/decaf/security/cert/CertificateException.h>
```

Inheritance diagram for `decaf::security::cert::CertificateException`:

### Public Member Functions

- **CertificateException** ( ) throw ( )  
*Default Constructor.*
- **CertificateException** (const Exception &ex) throw ( )  
*Conversion Constructor from some other Exception.*
- **CertificateException** (const **CertificateException** &ex) throw ( )  
*Copy Constructor.*
- **CertificateException** (const char \*file, const int lineNumber, const char \*msg,...) throw ( )  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **CertificateException** \* clone ( ) const

*Clones this exception.*

- virtual `~CertificateException () throw ()`

### 6.177.1 Constructor & Destructor Documentation

**6.177.1.1** `decaf::security::cert::CertificateException::CertificateException ( ) throw ()`  
`[inline]`

Default Constructor.

**6.177.1.2** `decaf::security::cert::CertificateException::CertificateException ( const Exception & ex ) throw ()` `[inline]`

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.177.1.3** `decaf::security::cert::CertificateException::CertificateException ( const CertificateException & ex ) throw ()` `[inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.177.1.4** `decaf::security::cert::CertificateException::CertificateException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
<i>...</i>	list of primitives that are formatted into the message

6.177.1.5 **virtual** `decaf::security::cert::CertificateException::~~CertificateException ( ) throw ( )`  
`[inline, virtual]`

## 6.177.2 Member Function Documentation

6.177.2.1 **virtual** `CertificateException*` `decaf::security::cert::CertificateException::clone ( ) const` `[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

A deep copy of this exception.

Reimplemented from `decaf::security::GeneralSecurityException` (p. 2037).

Reimplemented in `decaf::security::cert::CertificateEncodingException` (p. 1118),

`decaf::security::cert::CertificateExpiredException` (p. 1122), `decaf::security::cert::CertificateNotYetValidException` (p. 1124), and `decaf::security::cert::CertificateParsingException` (p. 1126).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateException.h`

## 6.178 `decaf::security::cert::CertificateExpiredException` Class Reference

```
#include <src/main/decaf/security/cert/CertificateExpiredException.h>
```

Inheritance diagram for `decaf::security::cert::CertificateExpiredException`:

### Public Member Functions

- **`CertificateExpiredException ( ) throw ( )`**  
*Default Constructor.*
- **`CertificateExpiredException (const Exception &ex) throw ( )`**  
*Conversion Constructor from some other Exception.*
- **`CertificateExpiredException (const CertificateExpiredException &ex) throw ( )`**  
*Copy Constructor.*



- **CertificateExpiredException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **CertificateExpiredException** \* clone () const  
*Clones this exception.*
- virtual ~**CertificateExpiredException** () throw ()

### 6.178.1 Constructor & Destructor Documentation

6.178.1.1 decaf::security::cert::CertificateExpiredException::CertificateExpiredException ( )  
throw () [inline]

Default Constructor.

6.178.1.2 decaf::security::cert::CertificateExpiredException::CertificateExpiredException ( const Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.178.1.3 decaf::security::cert::CertificateExpiredException::CertificateExpiredException ( const CertificateExpiredException & ex ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.178.1.4 decaf::security::cert::CertificateExpiredException::CertificateExpiredException ( const char \* file, const int lineNumber, const char \* msg, ... ) throw ()  
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.

<i>msg</i>	message to report
...	list of primitives that are formatted into the message

**6.178.1.5** `virtual decaf::security::cert::CertificateExpiredException::~CertificateExpiredException ( ) throw () [inline, virtual]`

## 6.178.2 Member Function Documentation

**6.178.2.1** `virtual CertificateExpiredException* decaf::security::cert::CertificateExpiredException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

A deep copy of this exception.

Reimplemented from `decaf::security::cert::CertificateException` (p. 1120).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateExpiredException.h`

## 6.179 decaf::security::cert::CertificateNotYetValidException Class Reference

```
#include <src/main/decaf/security/cert/CertificateNotYetValidException.h>
```

Inheritance diagram for `decaf::security::cert::CertificateNotYetValidException`:

### Public Member Functions

- **CertificateNotYetValidException ( ) throw ( )**  
*Default Constructor.*
- **CertificateNotYetValidException (const Exception &ex) throw ( )**  
*Conversion Constructor from some other Exception.*
- **CertificateNotYetValidException (const CertificateNotYetValidException &ex) throw ( )**

## 6.179 decaf::security::cert::CertificateNotYetValidException Class Reference 123

*Copy Constructor.*

- **CertificateNotYetValidException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()

*Constructor - Initializes the file name and line number where this message occurred.*

- virtual **CertificateNotYetValidException** \* clone () const

*Clones this exception.*

- virtual ~**CertificateNotYetValidException** () throw ()

### 6.179.1 Constructor & Destructor Documentation

6.179.1.1 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException ( ) throw () [inline]

Default Constructor.

6.179.1.2 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException ( const Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.179.1.3 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException ( const CertificateNotYetValidException & ex ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.179.1.4 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException ( const char \* file, const int lineNumber, const char \* msg, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
<i>...</i>	list of primitives that are formatted into the message

**6.179.1.5** `virtual decaf::security::cert::CertificateNotYetValidException::~~CertificateNotYetValidException ( ) throw () [inline, virtual]`

## 6.179.2 Member Function Documentation

**6.179.2.1** `virtual CertificateNotYetValidException* decaf::security::cert::CertificateNotYetValidException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

A deep copy of this exception.

Reimplemented from `decaf::security::cert::CertificateException` (p. 1120).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateNotYetValidException.h`

## 6.180 decaf::security::cert::CertificateParsingException Class Reference

```
#include <src/main/decaf/security/cert/CertificateParsingException.h>
```

Inheritance diagram for `decaf::security::cert::CertificateParsingException`:

### Public Member Functions

- **CertificateParsingException** ( ) throw ( )  
*Default Constructor.*
- **CertificateParsingException** (const Exception &ex) throw ( )  
*Conversion Constructor from some other Exception.*

- **CertificateParsingException** (const **CertificateParsingException** &ex) throw ()  
*Copy Constructor.*
- **CertificateParsingException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **CertificateParsingException** \* **clone** () const  
*Clones this exception.*
- virtual ~**CertificateParsingException** () throw ()

### 6.180.1 Constructor & Destructor Documentation

**6.180.1.1** decaf::security::cert::CertificateParsingException::CertificateParsingException ( ) throw () [inline]

Default Constructor.

**6.180.1.2** decaf::security::cert::CertificateParsingException::CertificateParsingException ( const Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.180.1.3** decaf::security::cert::CertificateParsingException::CertificateParsingException ( const CertificateParsingException & ex ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.180.1.4** decaf::security::cert::CertificateParsingException::CertificateParsingException ( const char \* file, const int lineNumber, const char \* msg, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
<i>...</i>	list of primitives that are formatted into the message

**6.180.1.5** `virtual decaf::security::cert::CertificateParsingException::~~CertificateParsingException ( ) throw () [inline, virtual]`

**6.180.2 Member Function Documentation**

**6.180.2.1** `virtual CertificateParsingException* decaf::security::cert::CertificateParsingException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns**

A deep copy of this exception.

Reimplemented from `decaf::security::cert::CertificateException` (p. 1120).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateParsingException.h`

**6.181 decaf::lang::Character Class Reference**

```
#include <src/main/decaf/lang/Character.h>
```

Inheritance diagram for `decaf::lang::Character`:

**Public Member Functions**

- **Character** (char value)
- virtual int **compareTo** (const **Character** &c) const  
*Compares this **Character** (p. 1126) instance with another.*
- virtual bool **operator==** (const **Character** &c) const  
*Compares equality between this object and the one passed.*

- virtual bool **operator<** (const **Character** &c) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual int **compareTo** (const char &c) const  
*Compares this **Character** (p. 1126) instance with a char type.*
- virtual bool **operator==** (const char &c) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const char &c) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- bool **equals** (const **Character** &c) const
- bool **equals** (const char &c) const
- std::string **toString** () const
- virtual double **doubleValue** () const  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const  
*Answers the float value which the receiver represents.*
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual short **shortValue** () const  
*Answers the short value which the receiver represents.*
- virtual int **intValue** () const  
*Answers the int value which the receiver represents.*
- virtual long long **longValue** () const  
*Answers the long value which the receiver represents.*

### Static Public Member Functions

- static **Character** **valueOf** (char value)  
*Returns a **Character** (p. 1126) instance representing the specified char value.*
- static bool **isWhitespace** (char c)  
*Indicates whether or not the given character is considered whitespace.*
- static bool **isDigit** (char c)

*Indicates whether or not the given character is a digit.*

- static bool **isLowerCase** (char c)

*Indicates whether or not the given character is a lower case character.*

- static bool **isUpperCase** (char c)

*Indicates whether or not the given character is a upper case character.*

- static bool **isLetter** (char c)

*Indicates whether or not the given character is a letter.*

- static bool **isLetterOrDigit** (char c)

*Indicates whether or not the given character is either a letter or a digit.*

- static bool **isISOControl** (char c)

*Answers whether the character is an ISO control character, which is a char that lays in the range of 0 to 1f and 7f to 9f.*

- static int **digit** (char c, int radix)

*Returns the numeric value of the character ch in the specified radix.*

## Static Public Attributes

- static const int **MIN\_RADIX** = 2

*The minimum radix available for conversion to and from strings.*

- static const int **MAX\_RADIX** = 36

*The maximum radix available for conversion to and from strings.*

- static const char **MIN\_VALUE** = (char)0x7F

*The minimum value that a signed char can take on.*

- static const char **MAX\_VALUE** = (char)0x80

*The maximum value that a signed char can take on.*

- static const int **SIZE** = 8

*The size of the primitive charactor in bits.*

## 6.181.1 Constructor & Destructor Documentation

### 6.181.1.1 decaf::lang::Character::Character ( char value )

#### Parameters

<i>value</i>	- char to wrap.
--------------	-----------------



## 6.181.2 Member Function Documentation

**6.181.2.1** `virtual unsigned char decaf::lang::Character::byteValue ( ) const [inline, virtual]`

Answers the byte value which the receiver represents.

### Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2919).

**6.181.2.2** `virtual int decaf::lang::Character::compareTo ( const char & c ) const [inline, virtual]`

Compares this **Character** (p. 1126) instance with a char type.

### Parameters

<i>c</i>	- the char instance to be compared
----------	------------------------------------

### Returns

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable< char >** (p. 1249).

**6.181.2.3** `virtual int decaf::lang::Character::compareTo ( const Character & c ) const [inline, virtual]`

Compares this **Character** (p. 1126) instance with another.

### Parameters

<i>c</i>	- the <b>Character</b> (p. 1126) instance to be compared
----------	--

### Returns

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable< Character >** (p. 1249).

**6.181.2.4 static int decaf::lang::Character::digit ( char *c*, int *radix* ) [static]**

Returns the numeric value of the character *ch* in the specified radix.

If the radix is not in the range `MIN_RADIX <= radix <= MAX_RADIX` or if the value of *ch* is not a valid digit in the specified radix, -1 is returned. A character is a valid digit if at least one of the following is true:

\* The method `isDigit` is true of the character and the single-character decomposition is less than the specified radix. In this case the decimal digit value is returned. \* The character is one of the uppercase Latin letters 'A' through 'Z' and its code is less than `radix + 'A' - 10`. In this case, `ch - 'A' + 10` is returned. \* The character is one of the lowercase Latin letters 'a' through 'z' and its code is less than `radix + 'a' - 10`. In this case, `ch - 'a' + 10` is returned.

**Parameters**

<i>c</i>	- the char to be converted
<i>radix</i>	- the radix of the number

**Returns**

the numeric value of the number represented in the given radix

**6.181.2.5 virtual double decaf::lang::Character::doubleValue ( ) const [inline, virtual]**

Answers the double value which the receiver represents.

**Returns**

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2919).

**6.181.2.6 bool decaf::lang::Character::equals ( const char & *c* ) const [inline, virtual]****Returns**

true if the two Characters have the same value.

Implements **decaf::lang::Comparable< char >** (p. 1250).

**6.181.2.7 bool decaf::lang::Character::equals ( const Character & *c* ) const [inline, virtual]****Returns**

true if the two **Character** (p. 1126) Objects have the same value.

Implements **decaf::lang::Comparable< Character >** (p. 1250).

**6.181.2.8** `virtual float decaf::lang::Character::floatValue ( ) const [inline, virtual]`

Answers the float value which the receiver represents.

#### Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2920).

**6.181.2.9** `virtual int decaf::lang::Character::intValue ( ) const [inline, virtual]`

Answers the int value which the receiver represents.

#### Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2920).

**6.181.2.10** `static bool decaf::lang::Character::isDigit ( char c ) [inline, static]`

Indicates whether or not the given character is a digit.

**6.181.2.11** `static bool decaf::lang::Character::isISOControl ( char c ) [inline, static]`

Answers whether the character is an ISO control character, which is a char that lays in the range of 0 to 1f and 7f to 9f.

#### Parameters

<i>c</i>	- the character, including supplementary characters
----------	---

#### Returns

true if the char is an ISO control character

**6.181.2.12** `static bool decaf::lang::Character::isLetter ( char c ) [inline, static]`

Indicates whether or not the given character is a letter.

**6.181.2.13** `static bool decaf::lang::Character::isLetterOrDigit ( char c ) [inline, static]`

Indicates whether or not the given character is either a letter or a digit.

**6.181.2.14** `static bool decaf::lang::Character::isLowerCase ( char c ) [inline, static]`

Indicates whether or not the given character is a lower case character.

**6.181.2.15** `static bool decaf::lang::Character::isUpperCase ( char c ) [inline, static]`

Indicates whether or not the given character is a upper case character.

**6.181.2.16** `static bool decaf::lang::Character::isWhitespace ( char c ) [inline, static]`

Indicates whether or not the given character is considered whitespace.

**6.181.2.17** `virtual long long decaf::lang::Character::longValue ( ) const [inline, virtual]`

Answers the long value which the receiver represents.

#### Returns

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2920).

**6.181.2.18** `virtual bool decaf::lang::Character::operator< ( const Character & c ) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>c</i>	- the value to be compared to this one.
----------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Character >** (p. 1250).

**6.181.2.19** `virtual bool decaf::lang::Character::operator< ( const char & c ) const`  
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<code>c</code>	- the value to be compared to this one.
----------------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< char >** (p. 1250).

**6.181.2.20** `virtual bool decaf::lang::Character::operator== ( const Character & c ) const`  
[inline, virtual]

Compares equality between this object and the one passed.

#### Parameters

<code>c</code>	- the value to be compared to this one.
----------------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Character >** (p. 1250).

**6.181.2.21** `virtual bool decaf::lang::Character::operator== ( const char & c ) const`  
[inline, virtual]

Compares equality between this object and the one passed.

#### Parameters

<code>c</code>	- the value to be compared to this one.
----------------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< char >** (p. 1250).

**6.181.2.22** `virtual short decaf::lang::Character::shortValue ( ) const` `[inline, virtual]`

Answers the short value which the receiver represents.

#### Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2920).

**6.181.2.23** `std::string decaf::lang::Character::toString ( ) const`

#### Returns

this **Character** (p. 1126) Object as a **String** (p. 3775) Representation

**6.181.2.24** `static Character decaf::lang::Character::valueOf ( char value )` `[inline, static]`

Returns a **Character** (p. 1126) instance representing the specified char value.

#### Parameters

<i>value</i>	- the primitive char to wrap.
--------------	-------------------------------

#### Returns

a new Charactor instance that wraps this value.

### 6.181.3 Field Documentation

**6.181.3.1** `const int decaf::lang::Character::MAX_RADIX = 36` `[static]`

The maximum radix available for conversion to and from strings.

**6.181.3.2** `const char decaf::lang::Character::MAX_VALUE = (char)0x80` `[static]`

The maximum value that a signed char can take on.

**6.181.3.3** `const int decaf::lang::Character::MIN_RADIX = 2` `[static]`

The minimum radix available for conversion to and from strings.

**6.181.3.4** `const char decaf::lang::Character::MIN_VALUE = (char)0x7F` [static]

The minimum value that a signed char can take on.

**6.181.3.5** `const int decaf::lang::Character::SIZE = 8` [static]

The size of the primitive character in bits.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Character.h`

**6.182 decaf::internal::nio::CharArrayBuffer Class Reference**

```
#include <src/main/decaf/internal/nio/CharArrayBuffer.h>
```

Inheritance diagram for decaf::internal::nio::CharArrayBuffer:

**Public Member Functions**

- **CharArrayBuffer** (int size, bool **readOnly**=false) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **CharArrayBuffer** (p. 1135) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **CharArrayBuffer** (char \*array, int size, int **offset**, int **length**, bool **readOnly**=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a **CharArrayBuffer** (p. 1135) object that wraps the given array.*
- **CharArrayBuffer** (const decaf::lang::Pointer< **ByteArrayAdapter** > &array, int **offset**, int **length**, bool **readOnly**=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.*
- **CharArrayBuffer** (const **CharArrayBuffer** &other)  
*Create a **CharArrayBuffer** (p. 1135) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*
- virtual **~CharArrayBuffer** ()
- virtual char \* **array** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

*Returns the character array that backs this buffer (optional operation).*

*Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.*

*Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.*

**Returns**

*the array that backs this **Buffer** (p. 936).*

**Exceptions**

<b>ReadOnlyBufferException</b> (p. 3260)	<i>if this <b>Buffer</b> (p. 936) is read only.</i>
<b>UnsupportedOperationException</b>	<i>if the underlying store has no array.</i>

- virtual int **arrayOffset** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*

*Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.*

**Returns**

*The offset into the backing array where index zero starts.*

**Exceptions**

<b>ReadOnlyBufferException</b> (p. 3260)	<i>if this <b>Buffer</b> (p. 936) is read only.</i>
<b>UnsupportedOperationException</b>	<i>if the underlying store has no array.</i>

- virtual CharBuffer \* **asReadOnlyBuffer** () const

*Creates a new, read-only char buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.*

*If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.*

*The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.*

**Returns**

*The new, read-only char buffer which the caller then owns.*

- virtual CharBuffer & **compact** () throw ( decaf::nio::ReadOnlyBufferException )

*Compacts this buffer.*

*The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 941) is copied to*



index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index `limit()` (p. 940) - 1 is copied to index  $n = \text{limit}() - 1 - p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

#### Returns

a reference to this **CharBuffer** (p. 1148).

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3260)	- If this buffer is read-only
---	-------------------------------

- virtual **CharBuffer** \* **duplicate** ()

Creates a new char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

a new char **Buffer** (p. 936) which the caller owns.

- virtual char **get** () throw ( decaf::nio::BufferUnderflowException )

Relative get method.

Reads the character at this buffer's current position, and then increments the position.

#### Returns

the char at the current position.

#### Exceptions

<b>BufferUnderflowException</b> (p. 967)	if there no more data to return
---	---------------------------------

- virtual char **get** (int index) const throw ( lang::exceptions::IndexOutOfBoundsException )

Absolute get method.

Reads the char at the given index.

#### Parameters

index	The index in the <b>Buffer</b> (p. 936) where the char is to be read.
-------	---

#### Returns

the char that is located at the given index.

#### Exceptions

<b>IndexOutOfBoundsException</b>	if index is not smaller than the buffer's limit or is negative.
----------------------------------	---

- virtual bool **hasArray** () const

*Tells whether or not this buffer is backed by an accessible char array.*

*If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.*

**Returns**

*true if, and only if, this buffer is backed by an array and is not read-only*

- virtual bool **isReadOnly** () const

*Tells whether or not this buffer is read-only.*

**Returns**

*true if, and only if, this buffer is read-only.*

- virtual CharBuffer & **put** (char value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes the given char into this buffer at the current position, and then increments the position.*

**Parameters**

value	<i>The char value to be written.</i>
-------	--------------------------------------

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 964)	<i>if this buffer's current position is not smaller than its limit</i>
<b>ReadOnlyBufferException</b> (p. 3260)	<i>if this buffer is read-only.</i>

- virtual CharBuffer & **put** (int index, char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes the given char into this buffer at the given index.*

**Parameters**

index	<i>The position in the <b>Buffer</b> (p. 936) to write the data.</i>
value	<i>The char to write.</i>

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>IndexOutOfBoundsException</b> (p. 3260)	<i>if index greater than the buffer's limit minus the size of the type being written, or index is negative.</i>
<b>ReadOnlyBufferException</b> (p. 3260)	<i>if this buffer is read-only.</i>

- virtual CharBuffer \* **slice** () const

Creates a new **CharBuffer** (p. 1148) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the newly create **CharBuffer** (p. 1148) which the caller owns.

- virtual lang::CharSequence \* **subSequence** (int start, int end) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.

The new buffer will share this buffer's content; that is, if the content of this buffer is mutable then modifications to one buffer will cause the other to be modified. The new buffer's capacity will be that of this buffer, its position will be **position()** (p. 941) + start, and its limit will be **position()** (p. 941) + end. The new **Buffer** (p. 936) will be read-only if, and only if, this buffer is read-only.

#### Parameters

start	The index, relative to the current position, of the first character in the subsequence; must be non-negative and no larger than <b>remaining()</b> (p. 941).
end	The index, relative to the current position, of the character following the last character in the subsequence; must be no smaller than start and no larger than <b>remaining()</b> (p. 941).

#### Returns

The new character buffer, caller owns.

#### Exceptions

IndexOutOfBoundsException	if the preconditions on start and end fail.
---------------------------	---

## Protected Member Functions

- virtual void **setReadOnly** (bool value)

Sets this **CharArrayBuffer** (p. 1135) as Read-Only.

## Protected Attributes

- bool **readOnly**
- decaf::lang::Pointer< ByteArrayAdapter > **\_array**
- int **offset**

- `int length`

### 6.182.1 Constructor & Destructor Documentation

**6.182.1.1** `decaf::internal::nio::CharArrayBuffer::CharArrayBuffer ( int size, bool readOnly = false ) throw ( decaf::lang::exceptions::IllegalArgumentException )`

Creates a **CharArrayBuffer** (p. 1135) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

#### Parameters

<i>size</i>	The size of the array, this is the limit we read and write to.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>IllegalArgumentException</i>	if the capacity value is negative.
---------------------------------	------------------------------------

**6.182.1.2** `decaf::internal::nio::CharArrayBuffer::CharArrayBuffer ( char * array, int size, int offset, int length, bool readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a **CharArrayBuffer** (p. 1135) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

#### Parameters

<i>array</i>	The actual array to wrap.
<i>size</i>	The size of the given array.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if offset is greater than array capacity.

**6.182.1.3** `decaf::internal::nio::CharArrayBuffer::CharArrayBuffer ( const decaf::lang::Pointer< ByteArrayAdapter > & array, int offset, int length, bool readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **CharArrayBuffer** (p. 1135) will be that of the remaining capacity of the passed buffer.

#### Parameters

<i>array</i>	The ByteArrayAdapter to wrap.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>NullPointerException</i>	if array is NULL
<i>IndexOutOfBoundsException</i>	if offset + length is greater than array size.

**6.182.1.4** `decaf::internal::nio::CharArrayBuffer::CharArrayBuffer ( const CharArrayBuffer & other )`

Create a **CharArrayBuffer** (p. 1135) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.

#### Parameters

<i>other</i>	The <b>CharArrayBuffer</b> (p. 1135) this one is to mirror.
--------------	---

**6.182.1.5** `virtual decaf::internal::nio::CharArrayBuffer::~~CharArrayBuffer ( ) [virtual]`

## 6.182.2 Member Function Documentation

**6.182.2.1** `virtual char* decaf::internal::nio::CharArrayBuffer::array ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Returns the character array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

the array that backs this **Buffer** (p. 936).

### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<b><i>UnsupportedOperationException</i></b>	if the underlying store has no array.

Implements **decaf::nio::CharBuffer** (p. 1154).

**6.182.2.2** `virtual int decaf::internal::nio::CharArrayBuffer::arrayOffset ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

The offset into the backing array where index zero starts.

### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<b><i>UnsupportedOperationException</i></b>	if the underlying store has no array.

Implements **decaf::nio::CharBuffer** (p. 1155).

**6.182.2.3** `virtual CharBuffer* decaf::internal::nio::CharArrayBuffer::asReadOnlyBuffer ( ) const [virtual]`

Creates a new, read-only char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

### Returns

The new, read-only char buffer which the caller then owns.

Implements **decaf::nio::CharBuffer** (p. 1155).

**6.182.2.4** `virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::compact ( ) throw ( decaf::nio::ReadOnlyBufferException )` [virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 941) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 940) - 1 is copied to index  $n = \text{limit}()$  (p. 940) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

### Returns

a reference to this **CharBuffer** (p. 1148).

### Exceptions

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	- If this buffer is read-only
--	-------------------------------

Implements **decaf::nio::CharBuffer** (p. 1156).

**6.182.2.5** `virtual CharBuffer* decaf::internal::nio::CharArrayBuffer::duplicate ( )` [virtual]

Creates a new char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns**

a new char **Buffer** (p. 936) which the caller owns.

Implements **decaf::nio::CharBuffer** (p. 1157).

**6.182.2.6** virtual char decaf::internal::nio::CharArrayBuffer::get ( ) throw ( decaf::nio::BufferUnderflowException ) [virtual]

Relative get method.

Reads the character at this buffer's current position, and then increments the position.

**Returns**

the char at the current position.

**Exceptions**

<i>BufferUnderflowException</i> (p. 967)	if there no more data to return
---	---------------------------------

Implements **decaf::nio::CharBuffer** (p. 1157).

**6.182.2.7** virtual char decaf::internal::nio::CharArrayBuffer::get ( int *index* ) const throw ( lang::exceptions::IndexOutOfBoundsException ) [virtual]

Absolute get method.

Reads the char at the given index.

**Parameters**

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the char is to be read.
--------------	---

**Returns**

the char that is located at the given index.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit or is negative.
----------------------------------	---

Implements **decaf::nio::CharBuffer** (p. 1157).



**6.182.2.8** `virtual bool decaf::internal::nio::CharArrayBuffer::hasArray ( ) const` `[inline, virtual]`

Tells whether or not this buffer is backed by an accessible char array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

#### Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::CharBuffer** (p. 1159).

**6.182.2.9** `virtual bool decaf::internal::nio::CharArrayBuffer::isReadOnly ( ) const` `[inline, virtual]`

Tells whether or not this buffer is read-only.

#### Returns

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 940).

**6.182.2.10** `virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::put ( char value ) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )` `[virtual]`

Writes the given char into this buffer at the current position, and then increments the position.

#### Parameters

<i>value</i>	The char value to be written.
--------------	-------------------------------

#### Returns

a reference to this buffer.

#### Exceptions

<b><i>BufferOverflowException</i></b> (p. 964)	if this buffer's current position is not smaller than its limit
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.

Implements **decaf::nio::CharBuffer** (p. 1161).

**6.182.2.11** `virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::put ( int index, char value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )` [virtual]

Writes the given char into this buffer at the given index.

#### Parameters

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data.
<i>value</i>	The char to write.

#### Returns

a reference to this buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<i>ReadOnlyBufferException</i> (p. 3260)	if this buffer is read-only.

Implements **decaf::nio::CharBuffer** (p. 1162).

**6.182.2.12** `virtual void decaf::internal::nio::CharArrayBuffer::setReadOnly ( bool value )`  
[inline, protected, virtual]

Sets this **CharArrayBuffer** (p. 1135) as Read-Only.

#### Parameters

<i>value</i>	Boolean value, true if this buffer is to be read-only, false otherwise.
--------------	---

**6.182.2.13** `virtual CharBuffer* decaf::internal::nio::CharArrayBuffer::slice ( ) const`  
[virtual]

Creates a new **CharBuffer** (p. 1148) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns**

the newly create **CharBuffer** (p. 1148) which the caller owns.

Implements **decaf::nio::CharBuffer** (p. 1164).

**6.182.2.14** `virtual lang::CharSequence* decaf::internal::nio::CharArrayBuffer::subSequence  
( int start, int end ) const throw ( de-  
caf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]`

Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.

The new buffer will share this buffer's content; that is, if the content of this buffer is mutable then modifications to one buffer will cause the other to be modified. The new buffer's capacity will be that of this buffer, its position will be **position()** (p. 941) + start, and its limit will be **position()** (p. 941) + end. The new **Buffer** (p. 936) will be read-only if, and only if, this buffer is read-only.

**Parameters**

<i>start</i>	The index, relative to the current position, of the first character in the subsequence; must be non-negative and no larger than <b>remaining()</b> (p. 941).
<i>end</i>	The index, relative to the current position, of the character following the last character in the subsequence; must be no smaller than start and no larger than <b>remaining()</b> (p. 941).

**Returns**

The new character buffer, caller owns.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if the preconditions on start and end fail.
----------------------------------	---

Implements **decaf::nio::CharBuffer** (p. 1165).

### 6.182.3 Field Documentation

- 6.182.3.1 **decaf::lang::Pointer<ByteArrayAdapter>**  
**decaf::internal::nio::CharArrayBuffer::\_array** [protected]
- 6.182.3.2 **int decaf::internal::nio::CharArrayBuffer::length** [protected]
- 6.182.3.3 **int decaf::internal::nio::CharArrayBuffer::offset** [protected]
- 6.182.3.4 **bool decaf::internal::nio::CharArrayBuffer::readOnly** [protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/CharArrayBuffer.h`

### 6.183 decaf::nio::CharBuffer Class Reference

This class defines four categories of operations upon character buffers:

```
#include <src/main/decaf/nio/CharBuffer.h>
```

Inheritance diagram for decaf::nio::CharBuffer:

#### Public Member Functions

- **virtual ~CharBuffer ()**
- **virtual std::string toString () const**
- **CharBuffer & append (char value) throw ( BufferOverflowException, ReadOnlyBufferException )**  
*Appends the specified character to this buffer.*
- **CharBuffer & append (const lang::CharSequence \*value) throw ( BufferOverflowException, ReadOnlyBufferException )**  
*Appends the specified character sequence to this buffer.*
- **CharBuffer & append (const lang::CharSequence \*value, int start, int end) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, BufferOverflowException, ReadOnlyBufferException )**  
*Appends a subsequence of the specified character sequence to this buffer If value is Null the the string "null" is appended to the buffer.*
- **virtual char \* array ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )**  
*Returns the character array that backs this buffer (optional operation).*

- virtual int **arrayOffset** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )  
*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*
- virtual **CharBuffer** \* **asReadOnlyBuffer** () const =0  
*Creates a new, read-only char buffer that shares this buffer's content.*
- char **charAt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads the character at the given index relative to the current position.*
- virtual **CharBuffer** & **compact** ()=0 throw ( ReadOnlyBufferException )  
*Compacts this buffer.*
- virtual **CharBuffer** \* **duplicate** ()=0  
*Creates a new char buffer that shares this buffer's content.*
- virtual char **get** ()=0 throw ( BufferUnderflowException )  
*Relative get method.*
- virtual char **get** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Absolute get method.*
- **CharBuffer** & **get** (std::vector< char > buffer) throw ( BufferUnderflowException )  
*Relative bulk get method.*
- **CharBuffer** & **get** (char \*buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Relative bulk get method.*
- virtual bool **hasArray** () const =0  
*Tells whether or not this buffer is backed by an accessible char array.*
- int **length** () const  
*Returns the length of this character buffer.*
- **CharBuffer** & **put** (**CharBuffer** &src) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )  
*This method transfers the chars remaining in the given source buffer into this buffer.*
- **CharBuffer** & **put** (const char \*buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*This method transfers chars into this buffer from the given source array.*

- **CharBuffer & put** (std::vector< char > &buffer) throw ( BufferOverflowException, ReadOnlyBufferException )

*This method transfers the entire content of the given source char array into this buffer.*

- virtual **CharBuffer & put** (char value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )

*Writes the given char into this buffer at the current position, and then increments the position.*

- virtual **CharBuffer & put** (int index, char value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )

*Writes the given char into this buffer at the given index.*

- **CharBuffer & put** (std::string &src, int start, int end) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Relative bulk put method (optional operation).*

- **CharBuffer & put** (const std::string &src) throw ( BufferOverflowException, ReadOnlyBufferException )

*Relative bulk put method (optional operation).*

- virtual int **read** (**CharBuffer** \*target) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, ReadOnlyBufferException )

*Attempts to read characters into the specified character buffer.*

- virtual **lang::CharSequence \* subSequence** (int start, int end) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.*

- virtual **CharBuffer \* slice** () const =0

*Creates a new **CharBuffer** (p. 1148) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **CharBuffer** &value) const

- virtual bool **equals** (const **CharBuffer** &value) const

- virtual bool **operator==** (const **CharBuffer** &value) const

- virtual bool **operator**< (const **CharBuffer** &value) const

### Static Public Member Functions

- static **CharBuffer** \* **allocate** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Allocates a new character buffer.*
- static **CharBuffer** \* **wrap** (char \*array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Wraps the passed buffer with a new **CharBuffer** (p. 1148).*
- static **CharBuffer** \* **wrap** (std::vector< char > &buffer)  
*Wraps the passed STL char Vector in a **CharBuffer** (p. 1148).*

### Protected Member Functions

- **CharBuffer** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **CharBuffer** (p. 1148) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

#### 6.183.1 Detailed Description

This class defines four categories of operations upon character buffers: o Absolute and relative get and put methods that read and write single characters; o Relative bulk get methods that transfer contiguous sequences of characters from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of characters from a character array, a string, or some other character buffer into this buffer. o Methods for compacting, duplicating, and slicing a character buffer.

Character buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing character array or string into a buffer, or by creating a view of an existing byte buffer

This class implements the CharSequence interface so that character buffers may be used wherever character sequences are accepted, for example in the regular-expression package decaf.util.regex.

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained. The sequence of statements

```
cb.put("text/"); cb.put(subtype); cb.put("; charset="); cb.put(enc);
```

can, for example, be replaced by the single statement

```
cb.put("text").put(subtype).put("; charset=").put(enc);
```

### 6.183.2 Constructor & Destructor Documentation

**6.183.2.1** `decaf::nio::CharBuffer::CharBuffer ( int capacity ) throw ( decaf::lang::exceptions::IllegalArgumentException )` [protected]

Creates a **CharBuffer** (p. 1148) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

#### Parameters

<i>capacity</i>	The size of the array, this is the limit we read and write to.
-----------------	--

#### Exceptions

<i>IllegalArgumentException</i>	if capacity is negative.
---------------------------------	--------------------------

**6.183.2.2** `virtual decaf::nio::CharBuffer::~~CharBuffer ( )` [inline, virtual]

### 6.183.3 Member Function Documentation

**6.183.3.1** `static CharBuffer* decaf::nio::CharBuffer::allocate ( int capacity ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [static]

Allocates a new character buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

#### Parameters

<i>capacity</i>	The size of the Char buffer in chars ( 1 byte ).
-----------------	--

#### Returns

the **CharBuffer** (p. 1148) that was allocated, caller owns.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if capacity is negative.
----------------------------------	--------------------------



**6.183.3.2 CharBuffer& decaf::nio::CharBuffer::append ( const lang::CharSequence \* *value*, int *start*, int *end* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, BufferOverflowException, ReadOnlyBufferException )** [virtual]

Appends a subsequence of the specified character sequence to this buffer If value is Null the the string "null" is appended to the buffer.

#### Parameters

<i>value</i>	The CharSequence to append.
<i>start</i>	The index to start appending from.
<i>end</i>	The index to append to.

#### Returns

a reference to this modified **CharBuffer** (p. 1148).

#### Exceptions

<b>BufferOverflowException</b> (p. 964)	if there is no more space
<b>ReadOnlyBufferException</b> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<b>IndexOutOfBoundsException</b>	if start > end, or > length of sequence.

Implements **decaf::lang::Appendable** (p. 734).

**6.183.3.3 CharBuffer& decaf::nio::CharBuffer::append ( char *value* ) throw ( BufferOverflowException, ReadOnlyBufferException )** [virtual]

Appends the specified character to this buffer.

#### Parameters

<i>value</i>	The char to append.
--------------	---------------------

#### Returns

a reference to this modified **CharBuffer** (p. 1148).

#### Exceptions

<b>BufferOverflowException</b> (p. 964)	if there is no more space
<b>ReadOnlyBufferException</b> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.

Implements **decaf::lang::Appendable** (p. 734).

**6.183.3.4 CharBuffer& decaf::nio::CharBuffer::append ( const lang::CharSequence \*  
value ) throw ( BufferOverflowException, ReadOnlyBufferException )  
[virtual]**

Appends the specified character sequence to this buffer.

If value is Null the the string "null" is appended to the buffer.

#### Parameters

<i>value</i>	The CharSequence to append.
--------------	-----------------------------

#### Returns

a reference to this modified **CharBuffer** (p. 1148)

#### Exceptions

<b>BufferOverflowEx- ception</b> (p. 964)	if there is no more space
<b>ReadOnlyBufferEx- ception</b> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.

Implements **decaf::lang::Appendable** (p. 735).

**6.183.3.5 virtual char\* decaf::nio::CharBuffer::array ( ) throw (**  
**decaf::lang::exceptions::UnsupportedOperationException,**  
**ReadOnlyBufferException ) [pure virtual]**

Returns the character array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

the array that backs this **Buffer** (p. 936).

#### Exceptions

<b>ReadOnlyBufferEx- ception</b> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<b>UnsupportedOpera- tionException</b>	if the underlying store has no array.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1141).

**6.183.3.6** `virtual int decaf::nio::CharBuffer::arrayOffset ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

The offset into the backing array where index zero starts.

#### Exceptions

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<i><b>UnsupportedOperationException</b></i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1142).

**6.183.3.7** `virtual CharBuffer* decaf::nio::CharBuffer::asReadOnlyBuffer ( ) const [pure virtual]`

Creates a new, read-only char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns

The new, read-only char buffer which the caller then owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1142).

**6.183.3.8** `char decaf::nio::CharBuffer::charAt ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Reads the character at the given index relative to the current position.

#### Parameters

<i>index</i>	- The index of the character to be read relative to position
--------------	--

#### Returns

The character at index **position()** (p. 941) + index.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if the index + the current position exceeds the size of the buffer or the index is negative.
----------------------------------	--

Implements **decaf::lang::CharSequence** (p. 1168).

**6.183.3.9** `virtual CharBuffer& decaf::nio::CharBuffer::compact ( ) throw ( ReadOnlyBufferException )` [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 941) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 940) - 1 is copied to index  $n = \text{limit}()$  (p. 940) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

#### Returns

a reference to this **CharBuffer** (p. 1148).

#### Exceptions

<i>ReadOnlyBufferException</i> (p. 3260)	- If this buffer is read-only
---	-------------------------------

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1143).

**6.183.3.10** `virtual int decaf::nio::CharBuffer::compareTo ( const CharBuffer & value ) const`  
[virtual]

**6.183.3.11** `virtual CharBuffer* decaf::nio::CharBuffer::duplicate ( )` [pure  
virtual]

Creates a new char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

a new char **Buffer** (p. 936) which the caller owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1143).

**6.183.3.12** `virtual bool decaf::nio::CharBuffer::equals ( const CharBuffer & value ) const`  
[virtual]

**6.183.3.13** `virtual char decaf::nio::CharBuffer::get ( ) throw ( BufferUnderflowException )`  
[pure virtual]

Relative get method.

Reads the character at this buffer's current position, and then increments the position.

#### Returns

the char at the current position.

#### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 967)	if there no more data to return
--	---------------------------------

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1144).

**6.183.3.14** `virtual char decaf::nio::CharBuffer::get ( int index ) const throw (`  
`decaf::lang::exceptions::IndexOutOfBoundsException )` [pure  
virtual]

Absolute get method.

Reads the char at the given index.

**Parameters**

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the char is to be read.
--------------	---

**Returns**

the char that is located at the given index.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit or is negative.
----------------------------------	---

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1144).

### 6.183.3.15 **CharBuffer& decaf::nio::CharBuffer::get ( std::vector< char > *buffer* ) throw ( BufferUnderflowException )**

Relative bulk get method.

This method transfers chars from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize( N )` before calling this get method.

**Returns**

a reference to this **CharBuffer** (p. 1148).

**Exceptions**

<i>BufferUnderflowException</i> (p. 967)	if there are fewer than length chars remaining in this buffer.
---	--

### 6.183.3.16 **CharBuffer& decaf::nio::CharBuffer::get ( char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( BufferUnderflowException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )**

Relative bulk get method.

This method transfers chars from this buffer into the given destination array. If there are fewer chars remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 941), then no bytes are transferred and a **BufferUnderflowException** (p. 967) is thrown.

Otherwise, this method copies length chars from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

**Parameters**

<i>buffer</i>	The pointer to an allocated buffer to fill.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The amount of data to put in the passed buffer.

**Returns**

a reference to this **Buffer** (p. 936).

**Exceptions**

<b><i>BufferUnderflowException</i></b> (p. 967)	if there are fewer than length chars remaining in this buffer
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

**6.183.3.17** `virtual bool decaf::nio::CharBuffer::hasArray ( ) const` [pure virtual]

Tells whether or not this buffer is backed by an accessible char array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns**

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1145).

**6.183.3.18** `int decaf::nio::CharBuffer::length ( ) const` [inline, virtual]

Returns the length of this character buffer.

**Returns**

the length of this buffer from the position to the limit.

Implements **decaf::lang::CharSequence** (p. 1168).

6.183.3.19 `virtual bool decaf::nio::CharBuffer::operator< ( const CharBuffer & value ) const`  
[virtual]

6.183.3.20 `virtual bool decaf::nio::CharBuffer::operator== ( const CharBuffer & value ) const`  
[virtual]

6.183.3.21 `CharBuffer& decaf::nio::CharBuffer::put ( CharBuffer & src )`  
`throw ( BufferOverflowException, ReadOnlyBufferException,`  
`decaf::lang::exceptions::IllegalArgumentException )`

This method transfers the chars remaining in the given source buffer into this buffer.

If there are more chars remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 941), then no chars are transferred and a **BufferOverflowException** (p. 964) is thrown.

Otherwise, this method copies `n = src.remaining()` chars from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

#### Parameters

<i>src</i>	- the buffer to take chars from an place in this one.
------------	---

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 964)	if there is insufficient space in this buffer for the remaining chars in the source buffer.
<b>IllegalArgumentException</b>	if the source buffer is this buffer.
<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only.

6.183.3.22 `CharBuffer& decaf::nio::CharBuffer::put ( const char * buffer, int size, int offset,`  
`int length ) throw ( BufferOverflowException, ReadOnlyBufferException,`  
`decaf::lang::exceptions::IndexOutOfBoundsException,`  
`decaf::lang::exceptions::NullPointerException )`

This method transfers chars into this buffer from the given source array.

If there are more chars to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 941), then no chars are transferred and a **BufferOverflowException** (p. 964) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of



this buffer is then incremented by length.

#### Parameters

<i>buffer</i>	The array from which chars are to be read.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The offset within the array of the first char to be read.
<i>length</i>	The number of chars to be read from the given array.

#### Returns

a reference to this buffer.

#### Exceptions

<b><i>BufferOverflowException</i></b> (p. 964)	if there is insufficient space in this buffer
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

**6.183.3.23** `virtual CharBuffer& decaf::nio::CharBuffer::put ( char value ) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes the given char into this buffer at the current position, and then increments the position.

#### Parameters

<i>value</i>	The char value to be written.
--------------	-------------------------------

#### Returns

a reference to this buffer.

#### Exceptions

<b><i>BufferOverflowException</i></b> (p. 964)	if this buffer's current position is not smaller than its limit
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1145).

**6.183.3.24** `virtual CharBuffer& decaf::nio::CharBuffer::put ( int index, char value )  
throw ( decaf::lang::exceptions::IndexOutOfBoundsException,  
ReadOnlyBufferException ) [pure virtual]`

Writes the given char into this buffer at the given index.

#### Parameters

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data.
<i>value</i>	The char to write.

#### Returns

a reference to this buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<i>ReadOnlyBufferException</i> (p. 3260)	if this buffer is read-only.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1146).

**6.183.3.25** `CharBuffer& decaf::nio::CharBuffer::put ( std::string & src, int start, int  
end ) throw ( BufferOverflowException, ReadOnlyBufferException,  
decaf::lang::exceptions::IndexOutOfBoundsException )`

Relative bulk put method (optional operation).

This method transfers characters from the given string into this buffer. If there are more characters to be copied from the string than remain in this buffer, that is, if `end - start > remaining()` (p. 941), then no characters are transferred and a **BufferOverflowException** (p. 964) is thrown.

#### Returns

a reference to this buffer

Otherwise, this method copies `n = end - start` characters from the given string into this buffer, starting at the given start index and at the current position of this buffer. The position of this buffer is then incremented by `n`.

#### Parameters

<i>src</i>	The string to copy from.
<i>start</i>	The position in <i>src</i> to start from.
<i>end</i>	The position in <i>src</i> to stop at.

**Returns**

a reference to this **CharBuffer** (p. 1148).

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 964)	if this buffer's current position is not
<b><i>IndexOutOfBoundsException</i></b>	if index greater than the buffer's limit minus the size of the type being written.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only

**6.183.3.26 CharBuffer& decaf::nio::CharBuffer::put ( const std::string & src ) throw ( BufferOverflowException, ReadOnlyBufferException )**

Relative bulk put method (optional operation).

This method transfers the entire content of the given source string into this buffer. An invocation of this method of the form dst.put(s) behaves in exactly the same way as the invocation.

**Parameters**

<i>src</i>	The string to copy from.
------------	--------------------------

**Returns**

a reference to this **CharBuffer** (p. 1148).

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 964)	if this buffer's current position is not.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.

**6.183.3.27 CharBuffer& decaf::nio::CharBuffer::put ( std::vector< char > & buffer ) throw ( BufferOverflowException, ReadOnlyBufferException )**

This method transfers the entire content of the given source char array into this buffer.

This is the same as calling put( &buffer[0], 0, buffer.size()).

**Parameters**

<i>buffer</i>	The buffer whose contents are copied to this <b>CharBuffer</b> (p. 1148).
---------------	---

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 964)	if there is insufficient space in this buffer.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.

```
6.183.3.28 virtual int decaf::nio::CharBuffer::read ( CharBuffer * target )
throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
ReadOnlyBufferException ) [virtual]
```

Attempts to read characters into the specified character buffer.

The buffer is used as a repository of characters as-is: the only changes made are the results of a put operation. No flipping or rewinding of the buffer is performed.

**Parameters**

<i>target</i>	The buffer to read characters into
---------------	------------------------------------

**Returns**

The number of characters added to the buffer, or string::npos if this source of characters is at its end

**Exceptions**

<b><i>NullPointerException</i></b>	if target is Null.
<b><i>IllegalArgumentException</i></b>	if target is this <b>CharBuffer</b> (p. 1148).
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is in read-only mode.

```
6.183.3.29 virtual CharBuffer* decaf::nio::CharBuffer::slice ( ) const [pure
virtual]
```

Creates a new **CharBuffer** (p. 1148) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers'

position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the newly create **CharBuffer** (p. 1148) which the caller owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1146).

**6.183.3.30** `virtual lang::CharSequence* decaf::nio::CharBuffer::subSequence ( int start, int end ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.

The new buffer will share this buffer's content; that is, if the content of this buffer is mutable then modifications to one buffer will cause the other to be modified. The new buffer's capacity will be that of this buffer, its position will be **position()** (p. 941) + start, and its limit will be **position()** (p. 941) + end. The new **Buffer** (p. 936) will be read-only if, and only if, this buffer is read-only.

### Parameters

<i>start</i>	The index, relative to the current position, of the first character in the subsequence; must be non-negative and no larger than <b>remaining()</b> (p. 941).
<i>end</i>	The index, relative to the current position, of the character following the last character in the subsequence; must be no smaller than start and no larger than <b>remaining()</b> (p. 941).

### Returns

The new character buffer, caller owns.

### Exceptions

<i>IndexOutOfBoundsException</i>	if the preconditions on start and end fail.
----------------------------------	---

Implements **decaf::lang::CharSequence** (p. 1168).

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1147).

**6.183.3.31** `virtual std::string decaf::nio::CharBuffer::toString ( ) const [virtual]`

### Returns

a std::string describing this object

Implements **decaf::lang::CharSequence** (p. 1169).

**6.183.3.32** `static CharBuffer* decaf::nio::CharBuffer::wrap ( char * array, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [static]

Wraps the passed buffer with a new **CharBuffer** (p. 1148).

The new buffer will be backed by the given char array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>array</i>	The array that will back the new buffer.
<i>size</i>	The size of the array passed in.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

#### Returns

a new **CharBuffer** (p. 1148) that is backed by buffer, caller owns.

#### Exceptions

<i>NullPointerException</i>	if the array pointer is Null.
<i>IndexOutOfBoundsException</i>	if capacity is negative.

**6.183.3.33** `static CharBuffer* decaf::nio::CharBuffer::wrap ( std::vector< char > & buffer )` [static]

Wraps the passed STL char Vector in a **CharBuffer** (p. 1148).

The new buffer will be backed by the given char array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).
---------------	--

#### Returns

a new **CharBuffer** (p. 1148) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/CharBuffer.h`

## 6.184 decaf::lang::CharSequence Class Reference

A **CharSequence** (p. 1167) is a readable sequence of char values.

```
#include <src/main/decaf/lang/CharSequence.h>
```

Inheritance diagram for decaf::lang::CharSequence:

### Public Member Functions

- virtual **~CharSequence** ()
- virtual int **length** () const =0
- virtual char **charAt** (int index) const =0 throw ( lang::exceptions::IndexOutOfBoundsException )

*Returns the Char at the specified index so long as the index is not greater than the length of the sequence.*

- virtual **CharSequence** \* **subSequence** (int start, int end) const =0 throw ( lang::exceptions::IndexOutOfBoundsException )

*Returns a new **CharSequence** (p. 1167) that is a subsequence of this sequence.*

- virtual std::string **toString** () const =0

### 6.184.1 Detailed Description

A **CharSequence** (p. 1167) is a readable sequence of char values. This interface provides uniform, read-only access to many different kinds of char sequences.

This interface does not define that a **CharSequence** (p. 1167) should implement comparable, it is therefore up to the derived classes that implement this interface to define equality, which implies that comparison of two CharSequences does not have a contract on equality.

## 6.184.2 Constructor & Destructor Documentation

6.184.2.1 `virtual decaf::lang::CharSequence::~~CharSequence ( ) [inline, virtual]`

## 6.184.3 Member Function Documentation

6.184.3.1 `virtual char decaf::lang::CharSequence::charAt ( int index ) const throw ( lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Returns the Char at the specified index so long as the index is not greater than the length of the sequence.

### Parameters

<i>index</i>	- position to return the char at.
--------------	-----------------------------------

### Returns

the char at the given position

### Exceptions

<i>IndexOutOfBoundsException</i>	if index is > than <b>length()</b> (p. 1168) or negative
----------------------------------	--

Implemented in **decaf::lang::String** (p. 3777), and **decaf::nio::CharBuffer** (p. 1156).

6.184.3.2 `virtual int decaf::lang::CharSequence::length ( ) const [pure virtual]`

### Returns

the length of the underlying character sequence.

Implemented in **decaf::lang::String** (p. 3778), and **decaf::nio::CharBuffer** (p. 1159).

6.184.3.3 `virtual CharSequence* decaf::lang::CharSequence::subSequence ( int start, int end ) const throw ( lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Returns a new **CharSequence** (p. 1167) that is a subsequence of this sequence.

The subsequence starts with the char value at the specified index and ends with the char value at index end - 1. The length (in chars) of the returned sequence is end - start, so if start == end then an empty sequence is returned.

### Parameters

<i>start</i>	- the start index, inclusive
<i>end</i>	- the end index, exclusive



**Returns**

a new **CharSequence** (p. 1167)

**Exceptions**

<i>IndexOutOfBoundsException</i>	if start or end > <b>length()</b> (p. 1168) or start or end are negative.
----------------------------------	---

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1147), **decaf::lang::String** (p. 3778), and **decaf::nio::CharBuffer** (p. 1165).

**6.184.3.4** `virtual std::string decaf::lang::CharSequence::toString ( ) const` [pure virtual]

**Returns**

the string representation of this **CharSequence** (p. 1167)

Implemented in **decaf::lang::String** (p. 3778), and **decaf::nio::CharBuffer** (p. 1165).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/CharSequence.h`

**6.185 decaf::util::zip::CheckedInputStream Class Reference**

An implementation of a **FilterInputStream** that will maintain a **Checksum** (p. 1174) of the bytes read, the **Checksum** (p. 1174) can then be used to verify the integrity of the input stream.

```
#include <src/main/decaf/util/zip/CheckedInputStream.h>
```

Inheritance diagram for **decaf::util::zip::CheckedInputStream**:

**Public Member Functions**

- **CheckedInputStream** (**InputStream** \***inputStream**, **Checksum** \***sum**, bool **own**=false)

*Create a new instance of a **CheckedInputStream** (p. 1169).*

- virtual **~CheckedInputStream** ()
- **Checksum** \* **getChecksum** () const

*Returns a Pointer to the **Checksum** (p. 1174) that is in use by this **CheckedInputStream** (p. 1169).*

- virtual long long **skip** (long long num) throw ( **decaf::io::IOException** )

*Skips over and discards  $n$  bytes of data from this input stream.*

*The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before  $n$  bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.*

*The skip method of **InputStream** (p. 2105) creates a byte array and then repeatedly reads into it until  $num$  bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*

#### Parameters

num	The number of bytes to skip.
-----	------------------------------

#### Returns

*total bytes skipped*

#### Exceptions

<b>IOException</b> (p. 2210)	<i>if an I/O error occurs.</i>
<b>UnsupportedOperationException</b>	<i>if the concrete stream class does not support skipping bytes.</i>

## Protected Member Functions

- virtual int **doReadByte** () throw ( decaf::io::IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

### 6.185.1 Detailed Description

An implementation of a **FilterInputStream** that will maintain a **Checksum** (p. 1174) of the bytes read, the **Checksum** (p. 1174) can then be used to verify the integrity of the input stream.

#### Since

1.0

### 6.185.2 Constructor & Destructor Documentation

#### 6.185.2.1 decaf::util::zip::CheckedInputStream::CheckedInputStream ( InputStream \* inputStream, Checksum \* sum, bool own = false )

Create a new instance of a **CheckedInputStream** (p. 1169).

#### Parameters

<i>inputStream</i>	The <b>InputStream</b> instance to Wrap.
<i>sum</i>	The <b>Checksum</b> (p. 1174) instance to use (does not take ownership of the Pointer).
<i>own</i>	Indicates if this filer should take ownership of the <b>InputStream</b> .

**Exceptions**

<i>NullPointerException</i>	if the <b>Checksum</b> (p. 1174) pointer is NULL.
-----------------------------	---

**6.185.2.2** virtual decaf::util::zip::ChecksumInputStream::~ChecksumInputStream ( )  
[virtual]

**6.185.3 Member Function Documentation**

**6.185.3.1** virtual int decaf::util::zip::ChecksumInputStream::doReadArrayBounded ( unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException ) [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1954).

**6.185.3.2** virtual int decaf::util::zip::ChecksumInputStream::doReadByte ( ) throw ( decaf::io::IOException ) [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1954).

**6.185.3.3** Checksum\* decaf::util::zip::ChecksumInputStream::getChecksum ( ) const  
[inline]

Returns a Pointer to the **Checksum** (p. 1174) that is in use by this **ChecksumInputStream** (p. 1169).

**Returns**

the pointer to the **Checksum** (p. 1174) instance that is in use by this object.

**6.185.3.4** virtual long long decaf::util::zip::ChecksumInputStream::skip ( long long *num* ) throw ( decaf::io::IOException ) [virtual]

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2105) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

**Parameters**

<i>num</i>	The number of bytes to skip.
------------	------------------------------

**Returns**

total bytes skipped

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
<b><i>UnsupportedOperationException</i></b>	if the concrete stream class does not support skipping bytes.

Adds the skipped bytes into the **Checksum** (p. 1174).

Reimplemented from **decaf::io::FilterInputStream** (p. 1956).

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**CheckedInputStream.h**

**6.186 decaf::util::zip::CheckedOutputStream Class Reference**

An implementation of a **FilterOutputStream** that will maintain a **Checksum** (p. 1174) of the bytes written, the **Checksum** (p. 1174) can then be used to verify the integrity of the output stream.

```
#include <src/main/decaf/util/zip/CheckedOutputStream.h>
```

Inheritance diagram for **decaf::util::zip::CheckedOutputStream**:

**Public Member Functions**

- **CheckedOutputStream** (**decaf::io::OutputStream** \*outputStream, **Checksum** \*sum, bool own=false)

Create a new instance of a **CheckedOutputStream** (p. 1172).

- virtual **~CheckedOutputStream** ()
- **Checksum** \* **getChecksum** () const

**Protected Member Functions**

- virtual void **doWriteByte** (unsigned char value) throw ( **decaf::io::IOException** )

- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### 6.186.1 Detailed Description

An implementation of a `FilterOutputStream` that will maintain a **Checksum** (p. 1174) of the bytes written, the **Checksum** (p. 1174) can then be used to verify the integrity of the output stream.

Since

1.0

### 6.186.2 Constructor & Destructor Documentation

6.186.2.1 `decaf::util::zip::CheckedOutputStream::CheckedOutputStream ( decaf::io::OutputStream * outputStream, Checksum * sum, bool own = false )`

Create a new instance of a **CheckedOutputStream** (p. 1172).

#### Parameters

<i>output-Stream</i>	The <code>OutputStream</code> instance to Wrap.
<i>sum</i>	The <b>Checksum</b> (p. 1174) instance to use (does not take ownership of the Pointer).
<i>own</i>	Indicates if this filer should take ownership of the <code>InputStream</code> .

#### Exceptions

<i>NullPointerException</i>	if the <b>Checksum</b> (p. 1174) pointer is NULL.
-----------------------------	---

6.186.2.2 `virtual decaf::util::zip::CheckedOutputStream::~~CheckedOutputStream ( )`  
[virtual]

### 6.186.3 Member Function Documentation

6.186.3.1 `virtual void decaf::util::zip::CheckedOutputStream::doWriteArrayBounded ( const unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`  
[protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1960).

6.186.3.2 `virtual void decaf::util::zip::CheckedOutputStream::doWriteByte ( unsigned char value ) throw ( decaf::io::IOException )` [protected, virtual]

Reimplemented from `decaf::io::FilterOutputStream` (p. 1960).

6.186.3.3 `Checksum* decaf::util::zip::CheckedOutputStream::getChecksum ( ) const` [inline]

#### Returns

a pointer to the **Checksum** (p. 1174) instance in use by this object.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/CheckedOutputStream.h`

## 6.187 decaf::util::zip::Checksum Class Reference

An interface used to represent **Checksum** (p. 1174) values in the Zip package.

```
#include <src/main/decaf/util/zip/Checksum.h>
```

Inheritance diagram for `decaf::util::zip::Checksum`:

#### Public Member Functions

- `virtual ~Checksum ()`
- `virtual long long getValue () const =0`
- `virtual void reset ()=0`  
*Reset the checksum to its initial value.*
- `virtual void update (const std::vector< unsigned char > &buffer)=0`  
*Updates the current checksum with the specified vector of bytes.*
- `virtual void update (const std::vector< unsigned char > &buffer, int offset, int length)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
*Updates the current checksum with the specified array of bytes.*
- `virtual void update (const unsigned char *buffer, int size, int offset, int length)=0 throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`  
*Updates the current checksum with the specified array of bytes.*
- `virtual void update (int byte)=0`  
*Updates the current checksum with the specified byte value.*

### 6.187.1 Detailed Description

An interface used to represent **Checksum** (p. 1174) values in the Zip package.

#### Since

1.0

### 6.187.2 Constructor & Destructor Documentation

**6.187.2.1** virtual decaf::util::zip::Checksum::~Checksum ( ) [inline, virtual]

### 6.187.3 Member Function Documentation

**6.187.3.1** virtual long long decaf::util::zip::Checksum::getValue ( ) const [pure virtual]

#### Returns

the current checksum value.

Implemented in **decaf::util::zip::Adler32** (p. 731), and **decaf::util::zip::CRC32** (p. 1569).

**6.187.3.2** virtual void decaf::util::zip::Checksum::reset ( ) [pure virtual]

Reset the checksum to its initial value.

Implemented in **decaf::util::zip::Adler32** (p. 731), and **decaf::util::zip::CRC32** (p. 1569).

**6.187.3.3** virtual void decaf::util::zip::Checksum::update ( const unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]

Updates the current checksum with the specified array of bytes.

#### Parameters

<i>buffer</i>	The buffer to read the updated bytes from.
<i>size</i>	The size of the passed buffer.
<i>offset</i>	The position in the buffer to start reading.
<i>length</i>	The amount of data to read from the byte buffer.

#### Exceptions

<i>NullPointerException</i>	if the passed buffer is NULL.
<i>IndexOutOfBoundsException</i>	if offset + length > size of the buffer.

Implemented in **decaf::util::zip::Adler32** (p. 731), and **decaf::util::zip::CRC32** (p. 1569).

**6.187.3.4** `virtual void decaf::util::zip::Checksum::update ( const std::vector< unsigned char > & buffer, int offset, int length ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Updates the current checksum with the specified array of bytes.

#### Parameters

<i>buffer</i>	The buffer to read the updated bytes from.
<i>offset</i>	The position in the buffer to start reading.
<i>length</i>	The amount of data to read from the byte buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if offset + length > size of the buffer.
----------------------------------	--

Implemented in **decaf::util::zip::Adler32** (p. 732), and **decaf::util::zip::CRC32** (p. 1570).

**6.187.3.5** `virtual void decaf::util::zip::Checksum::update ( const std::vector< unsigned char > & buffer ) [pure virtual]`

Updates the current checksum with the specified vector of bytes.

#### Parameters

<i>buffer</i>	The buffer to read the updated bytes from.
---------------	--

Implemented in **decaf::util::zip::Adler32** (p. 732), and **decaf::util::zip::CRC32** (p. 1570).

**6.187.3.6** `virtual void decaf::util::zip::Checksum::update ( int byte ) [pure virtual]`

Updates the current checksum with the specified byte value.

#### Parameters

<i>byte</i>	The byte value to update the current <b>Checksum</b> (p. 1174) with (0..255).
-------------	---

Implemented in **decaf::util::zip::Adler32** (p. 732), and **decaf::util::zip::CRC32** (p. 1570).

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/Checksum.h



## 6.188 decaf::lang::exceptions::ClassCastException Class Reference

```
#include <src/main/decaf/lang/exceptions/ClassCastException.h>
```

Inheritance diagram for decaf::lang::exceptions::ClassCastException:

### Public Member Functions

- **ClassCastException** () throw ()  
*Default Constructor.*
- **ClassCastException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1886).*
- **ClassCastException** (const **ClassCastException** &ex) throw ()  
*Copy Constructor.*
- **ClassCastException** (const std::exception \*cause) throw ()  
*Constructor.*
- **ClassCastException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **ClassCastException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **ClassCastException** \* **clone** () const  
*Clones this exception.*
- virtual ~**ClassCastException** () throw ()

### 6.188.1 Constructor & Destructor Documentation

**6.188.1.1** decaf::lang::exceptions::ClassCastException::ClassCastException ( ) throw ()  
[inline]

Default Constructor.

**6.188.1.2** decaf::lang::exceptions::ClassCastException::ClassCastException ( const **Exception** & ex ) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1886).

**Parameters**

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

**6.188.1.3** `decaf::lang::exceptions::ClassCastException::ClassCastException ( const  
ClassCastException & ex ) throw () [inline]`

Copy Constructor.

**Parameters**

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

**6.188.1.4** `decaf::lang::exceptions::ClassCastException::ClassCastException ( const  
std::exception * cause ) throw () [inline]`

Constructor.

**Parameters**

<i>cause</i>	<b>Pointer</b> (p. 3032) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

**6.188.1.5** `decaf::lang::exceptions::ClassCastException::ClassCastException ( const char * file,  
const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.188.1.6** `decaf::lang::exceptions::ClassCastException::ClassCastException ( const char * file,  
const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()  
[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
-------------	--------------------------------------

<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.188.1.7** `virtual decaf::lang::exceptions::ClassCastException::~~ClassCastException ( ) throw  
( ) [inline, virtual]`

## 6.188.2 Member Function Documentation

**6.188.2.1** `virtual ClassCastException* decaf::lang::exceptions::ClassCastException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

an new **Exception** (p. 1886) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1889).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/ClassCastException.h`

## 6.189 cms::Closeable Class Reference

Interface for a class that implements the close method.

```
#include <src/main/cms/Closeable.h>
```

Inheritance diagram for cms::Closeable:

### Public Member Functions

- `virtual ~Closeable ( )`
- `virtual void close ()=0 throw ( CMSException )`

*Closes this object and deallocates the appropriate resources.*

### 6.189.1 Detailed Description

Interface for a class that implements the close method. A class that implements this interface should release all resources upon the close call and should throw an exception from any methods that require those resources after they have been closed.

Since

1.0

### 6.189.2 Constructor & Destructor Documentation

6.189.2.1 `virtual cms::Closeable::~~Closeable ( ) [inline, virtual]`

### 6.189.3 Member Function Documentation

6.189.3.1 `virtual void cms::Closeable::close ( ) throw ( CMSException ) [pure virtual]`

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- If an error occurs while the resource is being closed.
---	--

Implemented in `activemq::cmsutil::CachedConsumer` (p. 1099), `activemq::cmsutil::CachedProducer` (p. 1103), `activemq::cmsutil::PooledSession` (p. 3045), `activemq::commands::ActiveMQTempDestination` (p. 580), `activemq::core::ActiveMQConnection` (p. 267), `activemq::core::ActiveMQConsumer` (p. 304), `activemq::core::ActiveMQProducer` (p. 468), `activemq::core::ActiveMQQueueBrowser` (p. 484), `activemq::core::ActiveMQSession` (p. 518), `cms::Connection` (p. 1298), and `cms::Session` (p. 3464).

The documentation for this class was generated from the following file:

- `src/main/cms/Closeable.h`

## 6.190 decaf::io::Closeable Class Reference

Interface for a class that implements the close method.

```
#include <src/main/decaf/io/Closeable.h>
```

Inheritance diagram for `decaf::io::Closeable`:

## Public Member Functions

- virtual `~Closeable()`
- virtual void `close()` throw ( `io::IOException` )

*Closes this object and deallocates the appropriate resources.*

### 6.190.1 Detailed Description

Interface for a class that implements the close method.

### 6.190.2 Constructor & Destructor Documentation

**6.190.2.1** virtual `decaf::io::Closeable::~~Closeable()` [`inline`, `virtual`]

### 6.190.3 Member Function Documentation

**6.190.3.1** virtual void `decaf::io::Closeable::close()` throw ( `io::IOException` ) [`pure virtual`]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an error occurs while closing.
--	-----------------------------------

Implemented in `activemq::transport::correlator::ResponseCorrelator` (p. 3386), `activemq::transport::failover::FailoverTransport` (p. 1934), `activemq::transport::inactivity::InactivityMonitor` (p. 2067), `activemq::transport::IOTransport` (p. 2215), `activemq::transport::mock::MockTransport` (p. 2857), `activemq::transport::tcp::TcpTransport` (p. 3867), `activemq::transport::TransportFilter` (p. 4007), `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p. 2987), `decaf::internal::io::StandardErrorOutputStream` (p. 3687), `decaf::internal::io::StandardOutputStream` (p. 3690), `decaf::internal::net::ssl::openssl::OpenSSLSocket` (p. 2947), `decaf::internal::net::ssl::openssl::OpenSSLSocketOutputSteam` (p. 2970), `decaf::internal::net::tcp::TcpSocketInputStream` (p. 3862), `decaf::internal::net::tcp::TcpSocketOutputStream` (p. 3864), `decaf::io::BlockingByteArrayInputStream` (p. 847), `decaf::io::BufferedInputStream` (p. 946), `decaf::io::FilterInputStream` (p. 1954), `decaf::io::FilterOutputStream` (p. 1959), `decaf::io::InputStream` (p. 2108), `decaf::io::InputStreamReader` (p. 2118), `decaf::io::OutputStream` (p. 2993), `decaf::io::OutputStreamWriter` (p. 3001), `decaf::net::Socket` (p. 3614), `decaf::util::logging::ConsoleHandler` (p. 1440), `decaf::util::logging::StreamHandler` (p. 3758), `decaf::util::zip::DeflaterOutputStream` (p. 1772), and `decaf::util::zip::InflaterInputStream` (p. 2102).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/Closeable.h`

## 6.191 activemq::transport::failover::CloseTransportsTask Class Reference

```
#include <src/main/activemq/transport/failover/CloseTransportsTask.h>
```

Inheritance diagram for `activemq::transport::failover::CloseTransportsTask`:

### Public Member Functions

- **CloseTransportsTask** ()
- virtual **~CloseTransportsTask** ()
- void **add** (const **Pointer**< **Transport** > &transport)  
*Add a new **Transport** (p. 3996) to close.*
- virtual bool **isPending** () const  
*This Task is pending if there are transports in the Queue that need to be closed.*
- virtual bool **iterate** ()  
*Return true until all transports have been closed and removed from the queue.*

### 6.191.1 Constructor & Destructor Documentation

6.191.1.1 `activemq::transport::failover::CloseTransportsTask::CloseTransportsTask ( )`

6.191.1.2 `virtual activemq::transport::failover::CloseTransportsTask::~~CloseTransportsTask ( )` [virtual]

### 6.191.2 Member Function Documentation

6.191.2.1 `void activemq::transport::failover::CloseTransportsTask::add ( const Pointer< Transport > & transport )`

Add a new **Transport** (p. 3996) to close.

6.191.2.2 `virtual bool activemq::transport::failover::CloseTransportsTask::isPending ( )` const [virtual]

This Task is pending if there are transports in the Queue that need to be closed.

### Returns

true if there is a transport in the queue that needs closed.

Implements `activemq::threads::CompositeTask` (p. 1255).

**6.191.2.3** `virtual bool activemq::transport::failover::CloseTransportsTask::iterate ( )`  
`[virtual]`

Return true until all transports have been closed and removed from the queue.

Implements **activemq::threads::Task** (p. 3847).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/CloseTransportsTask.h`

## 6.192 activemq::cmsutil::CmsAccessor Class Reference

Base class for **activemq.cmsutil.CmsTemplate** (p. 1201) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 1361) to operate on.

```
#include <src/main/activemq/cmsutil/CmsAccessor.h>
```

Inheritance diagram for `activemq::cmsutil::CmsAccessor`:

### Public Member Functions

- **CmsAccessor** ()
- `virtual ~CmsAccessor` ()
- `virtual ResourceLifecycleManager * getResourceLifecycleManager` ()
- `virtual const ResourceLifecycleManager * getResourceLifecycleManager` ()  
const
- `virtual void setConnectionFactory` (`cms::ConnectionFactory *connectionFactory`)

*Set the ConnectionFactory to use for obtaining CMS Connections.*

- `virtual const cms::ConnectionFactory * getConnectionFactory` () const  
*Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.*

- `virtual cms::ConnectionFactory * getConnectionFactory` ()  
*Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.*

- `virtual void setSessionAcknowledgeMode` (`cms::Session::AcknowledgeMode sessionAcknowledgeMode`)  
*Set the CMS acknowledgment mode that is used when creating a CMS Session to send a message.*

- `virtual cms::Session::AcknowledgeMode getSessionAcknowledgeMode` () const

*Return the acknowledgment mode for CMS sessions.*

### Protected Member Functions

- **CmsAccessor** (const **CmsAccessor** &)
- **CmsAccessor** & **operator=** (const **CmsAccessor** &)
- virtual void **init** () throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )  
*Initializes this object and prepares it for use.*
- virtual void **destroy** () throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )  
*Shuts down this object and destroys any allocated resources.*
- virtual **cms::Connection** \* **createConnection** () throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )  
*Create a CMS Connection via this template's ConnectionFactory.*
- virtual **cms::Session** \* **createSession** (**cms::Connection** \*con) throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )  
*Create a CMS Session for the given Connection.*
- virtual void **checkConnectionFactory** () throw ( decaf::lang::exceptions::IllegalStateException )  
*Verifies that the connection factory is valid.*

#### 6.192.1 Detailed Description

Base class for **activemq.cmsutil.CmsTemplate** (p. 1201) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 1361) to operate on. The subclass **activemq.cmsutil.CmsDestinationAccessor** (p. 1187) adds further, destination-related properties.

Not intended to be used directly.

#### See also

**activemq.cmsutil.CmsDestinationAccessor** (p. 1187)  
**activemq.cmsutil.CmsTemplate** (p. 1201)



## 6.192.2 Constructor & Destructor Documentation

6.192.2.1 `activemq::cmsutil::CmsAccessor::CmsAccessor ( const CmsAccessor & )`  
`[inline, protected]`

6.192.2.2 `activemq::cmsutil::CmsAccessor::CmsAccessor ( )`

6.192.2.3 `virtual activemq::cmsutil::CmsAccessor::~~CmsAccessor ( )` `[virtual]`

## 6.192.3 Member Function Documentation

6.192.3.1 `virtual void activemq::cmsutil::CmsAccessor::checkConnectionFactory ( ) throw ( decaf::lang::exceptions::IllegalStateException )` `[protected, virtual]`

Verifies that the connection factory is valid.

6.192.3.2 `virtual cms::Connection* activemq::cmsutil::CmsAccessor::createConnection ( ) throw ( cms::CMSEException, decaf::lang::exceptions::IllegalStateException )`  
`[protected, virtual]`

Create a CMS Connection via this template's ConnectionFactory.

### Returns

the new CMS Connection

### Exceptions

<i>cms::CMSEException</i> (p. 1190)	if thrown by CMS API methods
--	------------------------------

6.192.3.3 `virtual cms::Session* activemq::cmsutil::CmsAccessor::createSession ( cms::Connection * con ) throw ( cms::CMSEException, decaf::lang::exceptions::IllegalStateException )` `[protected, virtual]`

Create a CMS Session for the given Connection.

### Parameters

<i>con</i>	the CMS Connection to create a Session for
------------	--

### Returns

the new CMS Session

**Exceptions**

<b><i>cms::CMSException</i></b> (p. 1190)	if thrown by CMS API methods
--	------------------------------

**6.192.3.4** `virtual void activemq::cmsutil::CmsAccessor::destroy ( ) throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )`  
[inline, protected, virtual]

Shuts down this object and destroys any allocated resources.

Reimplemented in **activemq::cmsutil::CmsDestinationAccessor** (p. 1189), and **activemq::cmsutil::CmsTemplate** (p. 1205).

**6.192.3.5** `virtual const cms::ConnectionFactory* activemq::cmsutil::CmsAccessor::getConnectionFactory ( ) const` [inline, virtual]

Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.

**6.192.3.6** `virtual cms::ConnectionFactory* activemq::cmsutil::CmsAccessor::getConnectionFactory ( )`  
[inline, virtual]

Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.

**6.192.3.7** `virtual ResourceLifecycleManager* activemq::cmsutil::CmsAccessor::getResourceLifecycleManager ( )` [inline, virtual]

**6.192.3.8** `virtual const ResourceLifecycleManager* activemq::cmsutil::CmsAccessor::getResourceLifecycleManager ( ) const`  
[inline, virtual]

**6.192.3.9** `virtual cms::Session::AcknowledgeMode activemq::cmsutil::CmsAccessor::getSessionAcknowledgeMode ( ) const`  
[inline, virtual]

Return the acknowledgment mode for CMS sessions.

**Returns**

the acknowledgment mode applied by this accessor

**6.192.3.10** `virtual void activemq::cmsutil::CmsAccessor::init ( ) throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException ) [inline, protected, virtual]`

Initializes this object and prepares it for use.

This should be called before any other methods are called. This version does nothing.

Reimplemented in **activemq::cmsutil::CmsDestinationAccessor** (p. 1189), and **activemq::cmsutil::CmsTemplate** (p. 1207).

**6.192.3.11** `CmsAccessor& activemq::cmsutil::CmsAccessor::operator= ( const CmsAccessor & ) [inline, protected]`

**6.192.3.12** `virtual void activemq::cmsutil::CmsAccessor::setConnectionFactory ( cms::ConnectionFactory * connectionFactory ) [inline, virtual]`

Set the ConnectionFactory to use for obtaining CMS Connections.

**6.192.3.13** `virtual void activemq::cmsutil::CmsAccessor::setSessionAcknowledgeMode ( cms::Session::AcknowledgeMode sessionAcknowledgeMode ) [inline, virtual]`

Set the CMS acknowledgment mode that is used when creating a CMS Session to send a message.

Default is AUTO\_ACKNOWLEDGE.

#### Parameters

<i>sessionAcknowledgeMode</i>	the acknowledgment mode
-------------------------------	-------------------------

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsAccessor.h`

## 6.193 activemq::cmsutil::CmsDestinationAccessor Class Reference

Extends the **CmsAccessor** (p. 1183) to add support for resolving destination names.

```
#include <src/main/activemq/cmsutil/CmsDestinationAccessor.h>
```

Inheritance diagram for `activemq::cmsutil::CmsDestinationAccessor`:

## Public Member Functions

- **CmsDestinationAccessor** ()
- virtual **~CmsDestinationAccessor** ()
- virtual bool **isPubSubDomain** () const
- virtual void **setPubSubDomain** (bool pubSubDomain)
- virtual **DestinationResolver** \* **getDestinationResolver** ()
- virtual const **DestinationResolver** \* **getDestinationResolver** () const
- virtual void **setDestinationResolver** (**DestinationResolver** \*destRes)

## Protected Member Functions

- **CmsDestinationAccessor** (const **CmsDestinationAccessor** &)
- **CmsDestinationAccessor** & **operator=** (const **CmsDestinationAccessor** &)
- virtual void **init** () throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )  
*Initializes the destination resolver.*
- virtual void **destroy** () throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )  
*Calls **destroy**() (p. 1189) on the destination resolver.*
- virtual **cms::Destination** \* **resolveDestinationName** (**cms::Session** \*session, const std::string &destName) throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )  
*Resolves the destination via the **DestinationResolver** (p. 1810).*
- virtual void **checkDestinationResolver** () throw ( decaf::lang::exceptions::IllegalStateException )  
*Verifies that the destination resolver is valid.*

### 6.193.1 Detailed Description

Extends the **CmsAccessor** (p. 1183) to add support for resolving destination names. Not intended to be used directly.

#### See also

**CmsTemplate** (p. 1201)  
**CmsAccessor** (p. 1183)

### 6.193.2 Constructor & Destructor Documentation

6.193.2.1 `activemq::cmsutil::CmsDestinationAccessor::CmsDestinationAccessor ( const CmsDestinationAccessor & ) [inline, protected]`

6.193.2.2 `activemq::cmsutil::CmsDestinationAccessor::CmsDestinationAccessor ( )`

6.193.2.3 `virtual activemq::cmsutil::CmsDestinationAccessor::~~CmsDestinationAccessor ( ) [virtual]`

### 6.193.3 Member Function Documentation

6.193.3.1 `virtual void activemq::cmsutil::CmsDestinationAccessor::checkDestinationResolver ( ) throw ( decaf::lang::exceptions::IllegalStateException ) [protected, virtual]`

Verifies that the destination resolver is valid.

6.193.3.2 `virtual void activemq::cmsutil::CmsDestinationAccessor::destroy ( ) throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException ) [protected, virtual]`

Calls **destroy()** (p. 1189) on the destination resolver.

Reimplemented from **activemq::cmsutil::CmsAccessor** (p. 1186).

Reimplemented in **activemq::cmsutil::CmsTemplate** (p. 1205).

6.193.3.3 `virtual const DestinationResolver* activemq::cmsutil::CmsDestinationAccessor::getDestinationResolver ( ) const [inline, virtual]`

6.193.3.4 `virtual DestinationResolver* activemq::cmsutil::CmsDestinationAccessor::getDestinationResolver ( ) [inline, virtual]`

6.193.3.5 `virtual void activemq::cmsutil::CmsDestinationAccessor::init ( ) throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException ) [protected, virtual]`

Initializes the destination resolver.

Reimplemented from **activemq::cmsutil::CmsAccessor** (p. 1187).

Reimplemented in **activemq::cmsutil::CmsTemplate** (p. 1207).

6.193.3.6 `virtual bool activemq::cmsutil::CmsDestinationAccessor::isPubSubDomain ( ) const`  
`[inline, virtual]`

6.193.3.7 `CmsDestinationAccessor& activemq::cmsutil::CmsDestinationAccessor::operator= ( const`  
`CmsDestinationAccessor & ) [inline, protected]`

6.193.3.8 `virtual cms::Destination* activemq::cmsutil::CmsDestinationAccessor::resolveDestinationName`  
`( cms::Session * session, const std::string & destName ) throw (`  
`cms::CMSException, decaf::lang::exceptions::IllegalStateException )`  
`[protected, virtual]`

Resolves the destination via the `DestinationResolver` (p. 1810).

#### Parameters

<i>session</i>	the session
<i>destName</i>	the name of the destination.

#### Returns

the destination

#### Exceptions

<i>cms::CMSException</i> (p. 1190)	if resolution failed.
<i>decaf::lang::exceptions:</i> (p. 2060)	if the destination resolver property is NULL.

6.193.3.9 `virtual void activemq::cmsutil::CmsDestinationAccessor::setDestinationResolver (`  
`DestinationResolver * destRes ) [inline, virtual]`

6.193.3.10 `virtual void activemq::cmsutil::CmsDestinationAccessor::setPubSubDomain ( bool`  
`pubSubDomain ) [inline, virtual]`

Reimplemented in `activemq::cmsutil::CmsTemplate` (p. 1214).

Referenced by `activemq::cmsutil::CmsTemplate::setPubSubDomain()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsDestinationAccessor.h`

## 6.194 cms::CMSException Class Reference

CMS API Exception that is the base for all exceptions thrown from CMS classes.

```
#include <src/main/cms/CMSEException.h>
```

Inheritance diagram for cms::CMSEException:

## Public Member Functions

- **CMSEException** () throw ()
- **CMSEException** (const **CMSEException** &ex) throw ()
- **CMSEException** (const std::string &message, const std::exception \*cause) throw ()
- **CMSEException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**CMSEException** () throw ()
- virtual std::string **getMessage** () const  
*Gets the cause of the error.*
- virtual const std::exception \* **getCause** () const  
*Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.*
- virtual std::vector< std::pair< std::string, int > > **getStackTrace** () const  
*Provides the stack trace for every point where this exception was caught, marked, and rethrown.*
- virtual void **setMark** (const char \*file, const int lineNumber)  
*Adds a file/line number to the stack trace.*
- virtual void **printStackTrace** () const  
*Prints the stack trace to std::err.*
- virtual void **printStackTrace** (std::ostream &stream) const  
*Prints the stack trace to the given output stream.*
- virtual std::string **getStackTraceString** () const  
*Gets the stack trace as one contiguous string.*
- virtual const char \* **what** () const throw ()  
*Overloads the std::exception **what**() (p. 1194) function to return the cause of the exception.*

### 6.194.1 Detailed Description

CMS API Exception that is the base for all exceptions thrown from CMS classes. This class represents an error that has occurred in CMS, providers can wrap provider specific exceptions in this class by setting the cause to an instance of a provider specific exception provided it can be cast to an `std::exception`.

Since the contained cause exception is of type `std::exception` and the C++ exception class has no clone or copy method defined the contained exception can only be owned by one instance of an **CMSEException** (p. 1190). To that end the class hands off the exception to each successive copy so care must be taken when handling **CMSEException** (p. 1190) instances.

#### Since

1.0

### 6.194.2 Constructor & Destructor Documentation

6.194.2.1 `cms::CMSEException::CMSEException ( ) throw ()`

6.194.2.2 `cms::CMSEException::CMSEException ( const CMSEException & ex ) throw ()`

6.194.2.3 `cms::CMSEException::CMSEException ( const std::string & message, const std::exception * cause ) throw ()`

6.194.2.4 `cms::CMSEException::CMSEException ( const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace ) throw ()`

6.194.2.5 `virtual cms::CMSEException::~~CMSEException ( ) throw ()` [virtual]

### 6.194.3 Member Function Documentation

6.194.3.1 `virtual const std::exception* cms::CMSEException::getCause ( ) const` [virtual]

Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

#### Returns

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.



**6.194.3.2** virtual std::string cms::CMSException::getMessage ( ) const [virtual]

Gets the cause of the error.

#### Returns

string errors message

**6.194.3.3** virtual std::vector< std::pair< std::string, int> > cms::CMSException::getStackTrace ( ) const [virtual]

Provides the stack trace for every point where this exception was caught, marked, and rethrown.

#### Returns

vector containing stack trace strings

**6.194.3.4** virtual std::string cms::CMSException::getStackTraceString ( ) const [virtual]

Gets the stack trace as one contiguous string.

#### Returns

string with formatted stack trace data

**6.194.3.5** virtual void cms::CMSException::printStackTrace ( ) const [virtual]

Prints the stack trace to std::err.

**6.194.3.6** virtual void cms::CMSException::printStackTrace ( std::ostream & *stream* ) const [virtual]

Prints the stack trace to the given output stream.

#### Parameters

<i>stream</i>	the target output stream.
---------------	---------------------------

**6.194.3.7** virtual void cms::CMSException::setMark ( const char \* *file*, const int *lineNumber* ) [virtual]

Adds a file/line number to the stack trace.

**Parameters**

<i>file</i>	The name of the file calling this method (use <code>__FILE__</code> ).
<i>lineNumber</i>	The line number in the calling file (use <code>__LINE__</code> ).

**6.194.3.8 virtual const char\* cms::CMSException::what ( ) const throw () [virtual]**

Overloads the std::exception **what()** (p. 1194) function to return the cause of the exception.

**Returns**

const char pointer to error message

The documentation for this class was generated from the following file:

- `src/main/cms/CMSException.h`

**6.195 activemq::util::CMSExceptionSupport Class Reference**

```
#include <src/main/activemq/util/CMSExceptionSupport.h>
```

**Public Member Functions**

- virtual `~CMSExceptionSupport ()`

**Static Public Member Functions**

- static `cms::CMSException create` (const std::string &msg, const **decaf::lang::Exception** &cause)
- static `cms::CMSException create` (const **decaf::lang::Exception** &cause)
- static `cms::MessageEOFException createMessageEOFException` (const **decaf::lang::Exception** &cause)
- static `cms::MessageFormatException createMessageFormatException` (const **decaf::lang::Exception** &cause)

### 6.195.1 Constructor & Destructor Documentation

6.195.1.1 `virtual activemq::util::CMSExceptionSupport::~~CMSExceptionSupport ( )`  
[virtual]

### 6.195.2 Member Function Documentation

6.195.2.1 `static cms::CMSException activemq::util::CMSExceptionSupport::create ( const std::string & msg, const decaf::lang::Exception & cause )` [static]

6.195.2.2 `static cms::CMSException activemq::util::CMSExceptionSupport::create ( const decaf::lang::Exception & cause )` [static]

6.195.2.3 `static cms::MessageEOFException activemq::util::CMSExceptionSupport::createMessageEOFException ( const decaf::lang::Exception & cause )` [static]

6.195.2.4 `static cms::MessageFormatException activemq::util::CMSExceptionSupport::createMessageFormatException ( const decaf::lang::Exception & cause )` [static]

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getBooleanProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getByteProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getDoubleProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getFloatProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getIntProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getLongProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getShortProperty()`, and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getStringProperty()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/CMSExceptionSupport.h`

## 6.196 cms::CMSProperties Class Reference

Interface for a Java-like properties object.

```
#include <src/main/cms/CMSProperties.h>
```

Inheritance diagram for cms::CMSProperties:

### Public Member Functions

- `virtual ~CMSProperties ()`

- virtual bool **isEmpty** () const =0  
*Returns true if the properties object is empty.*
- virtual const char \* **getProperty** (const std::string &name) const =0  
*Looks up the value for the given property.*
- virtual std::string **getProperty** (const std::string &name, const std::string &defaultValue) const =0  
*Looks up the value for the given property.*
- virtual void **setProperty** (const std::string &name, const std::string &value)=0  
*Sets the value for a given property.*
- virtual bool **hasProperty** (const std::string &name) const =0  
*Check to see if the Property exists in the set.*
- virtual void **remove** (const std::string &name)=0  
*Removes the property with the given name.*
- virtual std::vector< std::pair< std::string, std::string > > **toArray** () const =0  
*Method that serializes the contents of the property map to an array.*
- virtual void **copy** (const **CMSPProperties** \*source)=0  
*Copies the contents of the given properties object to this one.*
- virtual **CMSPProperties** \* **clone** () const =0  
*Clones this object.*
- virtual void **clear** ()=0  
*Clears all properties from the map.*
- virtual std::string **toString** () const =0  
*Formats the contents of the Properties Object into a string that can be logged, etc.*

### 6.196.1 Detailed Description

Interface for a Java-like properties object. This is essentially a map of key-value string pairs.

Since

1.1

## 6.196.2 Constructor & Destructor Documentation

6.196.2.1 virtual cms::CMSProperties::~~CMSProperties ( ) [inline, virtual]

## 6.196.3 Member Function Documentation

6.196.3.1 virtual void cms::CMSProperties::clear ( ) [pure virtual]

Clears all properties from the map.

Implemented in `activemq::util::ActiveMQProperties` (p.476).

6.196.3.2 virtual CMSProperties\* cms::CMSProperties::clone ( ) const [pure virtual]

Clones this object.

### Returns

a replica of this object.

Implemented in `activemq::util::ActiveMQProperties` (p.476).

6.196.3.3 virtual void cms::CMSProperties::copy ( const CMSProperties \* *source* ) [pure virtual]

Copies the contents of the given properties object to this one.

### Parameters

<i>source</i>	The source properties object.
---------------	-------------------------------

6.196.3.4 virtual const char\* cms::CMSProperties::getProperty ( const std::string & *name* ) const [pure virtual]

Looks up the value for the given property.

### Parameters

<i>name</i>	The name of the property to be looked up.
-------------	---

### Returns

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

Implemented in `activemq::util::ActiveMQProperties` (p.477).

**6.196.3.5** `virtual std::string cms::CMSProperties::getProperty ( const std::string & name, const std::string & defaultValue ) const` [pure virtual]

Looks up the value for the given property.

#### Parameters

<i>name</i>	the name of the property to be looked up.
<i>defaultValue</i>	The value to be returned if the given property does not exist.

#### Returns

The value of the property specified by *name*, if it exists, otherwise the *defaultValue*.

Implemented in **activemq::util::ActiveMQProperties** (p. 477).

**6.196.3.6** `virtual bool cms::CMSProperties::hasProperty ( const std::string & name ) const` [pure virtual]

Check to see if the Property exists in the set.

#### Parameters

<i>name</i>	the name of the property to check
-------------	-----------------------------------

#### Returns

true if property exists, false otherwise.

Implemented in **activemq::util::ActiveMQProperties** (p. 477).

**6.196.3.7** `virtual bool cms::CMSProperties::isEmpty ( ) const` [pure virtual]

Returns true if the properties object is empty.

#### Returns

true if empty

Implemented in **activemq::util::ActiveMQProperties** (p. 478).

**6.196.3.8** `virtual void cms::CMSProperties::remove ( const std::string & name )` [pure virtual]

Removes the property with the given name.

#### Parameters

<i>name</i>	the name of the property to be removed.s
-------------	--

Implemented in **activemq::util::ActiveMQProperties** (p. 478).

**6.196.3.9** `virtual void cms::CMSProperties::setProperty ( const std::string & name, const std::string & value ) [pure virtual]`

Sets the value for a given property.

If the property already exists, overwrites the value.

#### Parameters

<i>name</i>	The name of the value to be written.
<i>value</i>	The value to be written.

Implemented in **activemq::util::ActiveMQProperties** (p. 478).

**6.196.3.10** `virtual std::vector< std::pair< std::string, std::string > > cms::CMSProperties::toArray ( ) const [pure virtual]`

Method that serializes the contents of the property map to an array.

#### Returns

list of pairs where the first is the name and the second is the value.

Implemented in **activemq::util::ActiveMQProperties** (p. 479).

**6.196.3.11** `virtual std::string cms::CMSProperties::toString ( ) const [pure virtual]`

Formats the contents of the Properties Object into a string that can be logged, etc.

#### Returns

string value of this object.

Implemented in **activemq::util::ActiveMQProperties** (p. 479).

The documentation for this class was generated from the following file:

- `src/main/cms/CMSProperties.h`

## 6.197 cms::CMSSecurityException Class Reference

This exception must be thrown when a provider rejects a user name/password submitted by a client.

```
#include <src/main/cms/CMSSecurityException.h>
```

Inheritance diagram for cms::CMSSecurityException:

### Public Member Functions

- **CMSSecurityException** () throw ()
- **CMSSecurityException** (const **CMSSecurityException** &ex) throw ()
- **CMSSecurityException** (const std::string &message, const std::exception \*cause) throw ()
- **CMSSecurityException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**CMSSecurityException** () throw ()

#### 6.197.1 Detailed Description

This exception must be thrown when a provider rejects a user name/password submitted by a client. It may also be thrown for any case where a security restriction prevents a method from completing.

Since

1.3

#### 6.197.2 Constructor & Destructor Documentation

6.197.2.1 cms::CMSSecurityException::CMSSecurityException ( ) throw ()

6.197.2.2 cms::CMSSecurityException::CMSSecurityException ( const CMSSecurityException & ex ) throw ()

6.197.2.3 cms::CMSSecurityException::CMSSecurityException ( const std::string & message, const std::exception \* cause ) throw ()

6.197.2.4 cms::CMSSecurityException::CMSSecurityException ( const std::string & message, const std::exception \* cause, const std::vector< std::pair< std::string, int > > & stackTrace ) throw ()

6.197.2.5 virtual cms::CMSSecurityException::~~CMSSecurityException ( ) throw ()  
[virtual]

The documentation for this class was generated from the following file:

- src/main/cms/CMSSecurityException.h



## 6.198 activemq::cmsutil::CmsTemplate Class Reference

**CmsTemplate** (p.1201) simplifies performing synchronous CMS operations.

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for activemq::cmsutil::CmsTemplate:

### Data Structures

- class **ProducerExecutor**
- class **ReceiveExecutor**
- class **ResolveProducerExecutor**
- class **ResolveReceiveExecutor**
- class **SendExecutor**

### Public Member Functions

- **CmsTemplate** ()
- **CmsTemplate** (cms::ConnectionFactory \*connectionFactory)
- virtual ~**CmsTemplate** ()
- virtual void **setDefaultDestination** (cms::Destination \*defaultDestination)  
*Sets the destination object to be used by default for send/receive operations.*
- virtual const cms::Destination \* **getDefaultDestination** () const  
*Retrieves the default destination to be used for send/receive operations.*
- virtual cms::Destination \* **getDefaultDestination** ()  
*Retrieves the default destination to be used for send/receive operations.*
- virtual void **setDefaultDestinationName** (const std::string &defaultDestinationName)  
*Sets the name of the default destination to be used from send/receive operations.*
- virtual const std::string **getDefaultDestinationName** () const  
*Gets the name of the default destination to be used for send/receive operations.*
- virtual void **setPubSubDomain** (bool pubSubDomain)  
*Indicates whether the default destination is a topic (true) or a queue (false).*
- virtual void **setMessageIdEnabled** (bool messageIdEnabled)
- virtual bool **isMessageIdEnabled** () const
- virtual void **setMessageTimestampEnabled** (bool messageTimestampEnabled)
- virtual bool **isMessageTimestampEnabled** () const
- virtual void **setNoLocal** (bool noLocal)

- virtual bool **isNoLocal** () const
- virtual void **setReceiveTimeout** (long long receiveTimeout)
- virtual long long **getReceiveTimeout** () const
- virtual void **setExplicitQosEnabled** (bool explicitQosEnabled)  
*Set if the QOS values (deliveryMode, priority, timeToLive) should be used for sending a message.*
- virtual bool **isExplicitQosEnabled** () const  
*If "true", then the values of deliveryMode, priority, and timeToLive will be used when sending a message.*
- virtual void **setDeliveryPersistent** (bool deliveryPersistent)  
*Set whether message delivery should be persistent or non-persistent, specified as boolean value ("true" or "false").*
- virtual void **setDeliveryMode** (int deliveryMode)  
*Set the delivery mode to use when sending a message.*
- virtual int **getDeliveryMode** () const  
*Return the delivery mode to use when sending a message.*
- virtual void **setPriority** (int priority)  
*Set the priority of a message when sending.*
- virtual int **getPriority** () const  
*Return the priority of a message when sending.*
- virtual void **setTimeToLive** (long long timeToLive)  
*Set the time-to-live of the message when sending.*
- virtual long long **getTimeToLive** () const  
*Return the time-to-live of the message when sending.*
- virtual void **execute** (**SessionCallback** \*action) throw ( cms::CMSException )  
*Executes the given action within a CMS Session.*
- virtual void **execute** (**ProducerCallback** \*action) throw ( cms::CMSException )  
*Executes the given action and provides it with a CMS Session and producer.*
- virtual void **execute** (cms::Destination \*dest, **ProducerCallback** \*action) throw ( cms::CMSException )  
*Executes the given action and provides it with a CMS Session and producer.*
- virtual void **execute** (const std::string &destinationName, **ProducerCallback** \*action) throw ( cms::CMSException )  
*Executes the given action and provides it with a CMS Session and producer.*

- virtual void **send** (**MessageCreator** \*messageCreator) throw ( cms::CMSEException )  
*Convenience method for sending a message to the default destination.*
- virtual void **send** (cms::Destination \*dest, **MessageCreator** \*messageCreator) throw ( cms::CMSEException )  
*Convenience method for sending a message to the specified destination.*
- virtual void **send** (const std::string &destinationName, **MessageCreator** \*messageCreator) throw ( cms::CMSEException )  
*Convenience method for sending a message to the specified destination.*
- virtual **cms::Message** \* **receive** () throw ( cms::CMSEException )  
*Performs a synchronous read from the default destination.*
- virtual **cms::Message** \* **receive** (cms::Destination \*destination) throw ( cms::CMSEException )  
*Performs a synchronous read from the specified destination.*
- virtual **cms::Message** \* **receive** (const std::string &destinationName) throw ( cms::CMSEException )  
*Performs a synchronous read from the specified destination.*
- virtual **cms::Message** \* **receiveSelected** (const std::string &selector) throw ( cms::CMSEException )  
*Performs a synchronous read consuming only messages identified by the given selector.*
- virtual **cms::Message** \* **receiveSelected** (cms::Destination \*destination, const std::string &selector) throw ( cms::CMSEException )  
*Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.*
- virtual **cms::Message** \* **receiveSelected** (const std::string &destinationName, const std::string &selector) throw ( cms::CMSEException )  
*Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.*

### Static Public Attributes

- static const long long **RECEIVE\_TIMEOUT\_NO\_WAIT**  
*Timeout value indicating that a receive operation should check if a message is immediately available without blocking.*
- static const long long **RECEIVE\_TIMEOUT\_INDEFINITE\_WAIT**

*Timeout value indicating a blocking receive without timeout.*

- static const int **DEFAULT\_PRIORITY**

*Default message priority.*

- static const long long **DEFAULT\_TIME\_TO\_LIVE**

*My default, messages should live forever.*

### Protected Member Functions

- **CmsTemplate** (const **CmsTemplate** &)
- **CmsTemplate** & **operator=** (const **CmsTemplate** &)
- void **init** () throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )  
*Initializes this object and prepares it for use.*
- void **destroy** () throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )  
*Clears all internal resources.*

### Friends

- class **ProducerExecutor**
- class **ResolveProducerExecutor**
- class **SendExecutor**
- class **ReceiveExecutor**
- class **ResolveReceiveExecutor**

### 6.198.1 Detailed Description

**CmsTemplate** (p. 1201) simplifies performing synchronous CMS operations. This class is intended to be for CMS what Spring's `JmsTemplate` is for JMS. Provided with a `CMS ConnectionFactory`, creates and manages all other CMS resources internally.

Before using **CmsTemplate** (p. 1201) the user must first set the destination (either by name or by setting the destination object directly) and then call `init` to initialize the object for use.

**CmsTemplate** (p. 1201) allows the user to get access to a `CMS Session` through a user-defined **SessionCallback** (p. 3475). Similarly, if the user wants direct access to a `CMS MessageProducer`, it can provide a **ProducerCallback** (p. 3154). As a convenience, the user can bypass having to provide callbacks altogether for sending messages, by calling one of the `send` methods.

## See also

**SessionCallback** (p. 3475)  
**ProducerCallback** (p. 3154)  
**MessageCreator** (p. 2677)

## 6.198.2 Constructor &amp; Destructor Documentation

6.198.2.1 **activemq::cmsutil::CmsTemplate::CmsTemplate ( const CmsTemplate & )**  
 [inline, protected]

6.198.2.2 **activemq::cmsutil::CmsTemplate::CmsTemplate ( )**

6.198.2.3 **activemq::cmsutil::CmsTemplate::CmsTemplate ( cms::ConnectionFactory \*  
 connectionFactory )**

6.198.2.4 **virtual activemq::cmsutil::CmsTemplate::~~CmsTemplate ( )** [virtual]

## 6.198.3 Member Function Documentation

6.198.3.1 **void activemq::cmsutil::CmsTemplate::destroy ( ) throw ( cms::CMSException,  
 decaf::lang::exceptions::IllegalStateException )** [protected,  
 virtual]

Clears all internal resources.

Reimplemented from **activemq::cmsutil::CmsDestinationAccessor** (p. 1189).

6.198.3.2 **virtual void activemq::cmsutil::CmsTemplate::execute ( SessionCallback \* action )**  
 throw ( cms::CMSException ) [virtual]

Executes the given action within a CMS Session.

## Parameters

<i>action</i>	the action to perform within a CMS Session
---------------	--

## Exceptions

<b><i>cms::CMSException</i></b> (p. 1190)	thrown if an error occurs.
--	----------------------------

6.198.3.3 **virtual void activemq::cmsutil::CmsTemplate::execute ( ProducerCallback \*  
 action ) throw ( cms::CMSException )** [virtual]

Executes the given action and provides it with a CMS Session and producer.

**Parameters**

<i>action</i>	the action to perform
---------------	-----------------------

**Exceptions**

<b><i>cms::CMSEException</i></b> (p. 1190)	thrown if an error occurs.
---	----------------------------

**6.198.3.4** `virtual void activemq::cmsutil::CmsTemplate::execute ( cms::Destination * dest,  
ProducerCallback * action ) throw ( cms::CMSEException ) [virtual]`

Executes the given action and provides it with a CMS Session and producer.

**Parameters**

<i>dest</i>	the destination to send messages to
<i>action</i>	the action to perform

**Exceptions**

<b><i>cms::CMSEException</i></b> (p. 1190)	thrown if an error occurs.
---	----------------------------

**6.198.3.5** `virtual void activemq::cmsutil::CmsTemplate::execute ( const std::string &  
destinationName, ProducerCallback * action ) throw ( cms::CMSEException )  
[virtual]`

Executes the given action and provides it with a CMS Session and producer.

**Parameters**

<i>destination-Name</i>	the name of the destination to send messages to (to internally be resolved to an actual destination)
<i>action</i>	the action to perform

**Exceptions**

<b><i>cms::CMSEException</i></b> (p. 1190)	thrown if an error occurs.
---	----------------------------

**6.198.3.6** `virtual cms::Destination* activemq::cmsutil::CmsTemplate::getDefaultDestination ( ) [inline, virtual]`

Retrieves the default destination to be used for send/receive operations.

**Returns**

the default destination. Non-const version of this method.

**6.198.3.7** `virtual const cms::Destination* activemq::cmsutil::CmsTemplate::getDefaultDestination ( ) const`  
[inline, virtual]

Retrieves the default destination to be used for send/receive operations.

#### Returns

the default destination. Const version of this method.

**6.198.3.8** `virtual const std::string activemq::cmsutil::CmsTemplate::getDefaultDestinationName ( ) const` [inline, virtual]

Gets the name of the default destination to be used for send/receive operations.

The destination type (topic/queue) is determined by the `pubSubDomain` property.

#### Returns

the default name of the destination for send/receive operations.

**6.198.3.9** `virtual int activemq::cmsutil::CmsTemplate::getDeliveryMode ( ) const`  
[inline, virtual]

Return the delivery mode to use when sending a message.

**6.198.3.10** `virtual int activemq::cmsutil::CmsTemplate::getPriority ( ) const` [inline, virtual]

Return the priority of a message when sending.

**6.198.3.11** `virtual long long activemq::cmsutil::CmsTemplate::getReceiveTimeout ( ) const`  
[inline, virtual]

**6.198.3.12** `virtual long long activemq::cmsutil::CmsTemplate::getTimeToLive ( ) const`  
[inline, virtual]

Return the time-to-live of the message when sending.

**6.198.3.13** `void activemq::cmsutil::CmsTemplate::init ( ) throw ( cms::CMSEException, decaf::lang::exceptions::IllegalStateException )` [protected, virtual]

Initializes this object and prepares it for use.

This should be called before any other methods are called.

Reimplemented from `activemq::cmsutil::CmsDestinationAccessor` (p. 1189).

**6.198.3.14** `virtual bool activemq::cmsutil::CmsTemplate::isExplicitQosEnabled ( ) const`  
`[inline, virtual]`

If "true", then the values of `deliveryMode`, `priority`, and `timeToLive` will be used when sending a message.

Otherwise, the default values, that may be set administratively, will be used.

#### Returns

true if overriding default values of QOS parameters (`deliveryMode`, `priority`, and `timeToLive`)

#### See also

`setDeliveryMode` (p. 1212)

`setPriority` (p. 1213)

`setTimeToLive` (p. 1214)

**6.198.3.15** `virtual bool activemq::cmsutil::CmsTemplate::isMessageIdEnabled ( ) const`  
`[inline, virtual]`

**6.198.3.16** `virtual bool activemq::cmsutil::CmsTemplate::isMessageTimestampEnabled ( )`  
`const [inline, virtual]`

**6.198.3.17** `virtual bool activemq::cmsutil::CmsTemplate::isNoLocal ( ) const` `[inline, virtual]`

**6.198.3.18** `CmsTemplate& activemq::cmsutil::CmsTemplate::operator= ( const`  
`CmsTemplate & ) [inline, protected]`

**6.198.3.19** `virtual cms::Message* activemq::cmsutil::CmsTemplate::receive ( ) throw (`  
`cms::CMSEException ) [virtual]`

Performs a synchronous read from the default destination.

#### Returns

the message

#### Exceptions

<i><b>cms::CMSEException</b></i> (p. 1190)	thrown if an error occurs
---	---------------------------



**6.198.3.20** `virtual cms::Message* activemq::cmsutil::CmsTemplate::receive ( const std::string & destinationName ) throw ( cms::CMSEException ) [virtual]`

Performs a synchronous read from the specified destination.

#### Parameters

<i>destination-Name</i>	the name of the destination to receive on (will be resolved to destination internally).
-------------------------	---

#### Returns

the message

#### Exceptions

<i>cms::CMSEException</i> (p. 1190)	thrown if an error occurs
--	---------------------------

**6.198.3.21** `virtual cms::Message* activemq::cmsutil::CmsTemplate::receive ( cms::Destination * destination ) throw ( cms::CMSEException ) [virtual]`

Performs a synchronous read from the specified destination.

#### Parameters

<i>destination</i>	the destination to receive on
--------------------	-------------------------------

#### Returns

the message

#### Exceptions

<i>cms::CMSEException</i> (p. 1190)	thrown if an error occurs
--	---------------------------

**6.198.3.22** `virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected ( const std::string & destinationName, const std::string & selector ) throw ( cms::CMSEException ) [virtual]`

Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

#### Parameters

<i>destination-Name</i>	the name of the destination to receive on (will be resolved to destination internally).
<i>selector</i>	the selector expression.

**Returns**

the message

**Exceptions**

<b><i>cms::CMSEException</i></b> (p. 1190)	thrown if an error occurs
---	---------------------------

**6.198.3.23** `virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected ( cms::Destination * destination, const std::string & selector ) throw ( cms::CMSEException )` [virtual]

Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

**Parameters**

<i>destination</i>	the destination to receive on.
<i>selector</i>	the selector expression.

**Returns**

the message

**Exceptions**

<b><i>cms::CMSEException</i></b> (p. 1190)	thrown if an error occurs
---	---------------------------

**6.198.3.24** `virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected ( const std::string & selector ) throw ( cms::CMSEException )` [virtual]

Performs a synchronous read consuming only messages identified by the given selector.

**Parameters**

<i>selector</i>	the selector expression.
-----------------	--------------------------

**Returns**

the message

**Exceptions**

<b><i>cms::CMSEException</i></b> (p. 1190)	thrown if an error occurs
---	---------------------------

**6.198.3.25** `virtual void activemq::cmsutil::CmsTemplate::send ( const std::string & destinationName, MessageCreator * messageCreator ) throw ( cms::CMSEException ) [virtual]`

Convenience method for sending a message to the specified destination.

#### Parameters

<i>destinationName</i>	The name of the destination to send to.
<i>messageCreator</i>	Responsible for creating the message to be sent

#### Exceptions

<b><i>cms::CMSEException</i></b> (p. 1190)	thrown if an error occurs.
---	----------------------------

**6.198.3.26** `virtual void activemq::cmsutil::CmsTemplate::send ( MessageCreator * messageCreator ) throw ( cms::CMSEException ) [virtual]`

Convenience method for sending a message to the default destination.

#### Parameters

<i>messageCreator</i>	Responsible for creating the message to be sent
-----------------------	---

#### Exceptions

<b><i>cms::CMSEException</i></b> (p. 1190)	thrown if an error occurs.
---	----------------------------

**6.198.3.27** `virtual void activemq::cmsutil::CmsTemplate::send ( cms::Destination * dest, MessageCreator * messageCreator ) throw ( cms::CMSEException ) [virtual]`

Convenience method for sending a message to the specified destination.

#### Parameters

<i>dest</i>	The destination to send to
<i>messageCreator</i>	Responsible for creating the message to be sent

#### Exceptions

<b><i>cms::CMSEException</i></b> (p. 1190)	thrown if an error occurs.
---	----------------------------

**6.198.3.28** `virtual void activemq::cmsutil::CmsTemplate::setDefaultDestination ( cms::Destination * defaultDestination ) [inline, virtual]`

Sets the destination object to be used by default for send/receive operations.

If no default destination is provided, the `defaultDestinationName` property is used to resolve this default destination for send/receive operations.

#### Parameters

<i>defaultDestination</i>	the default destination
---------------------------	-------------------------

**6.198.3.29** `virtual void activemq::cmsutil::CmsTemplate::setDefaultDestinationName ( const std::string & defaultDestinationName ) [inline, virtual]`

Sets the name of the default destination to be used from send/receive operations.

Calling this method will set the `defaultDestination` property to NULL. The destination type (topic/queue) is determined by the `pubSubDomain` property.

#### Parameters

<i>defaultDestinationName</i>	the name of the destination for send/receive to by default.
-------------------------------	---

**6.198.3.30** `virtual void activemq::cmsutil::CmsTemplate::setDeliveryMode ( int deliveryMode ) [inline, virtual]`

Set the delivery mode to use when sending a message.

Default is the Message default: "PERSISTENT".

Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

#### Parameters

<i>deliveryMode</i>	the delivery mode to use
---------------------	--------------------------

#### See also

**isExplicitQosEnabled** (p. 1208)

**6.198.3.31** `virtual void activemq::cmsutil::CmsTemplate::setDeliveryPersistent ( bool deliveryPersistent ) [inline, virtual]`

Set whether message delivery should be persistent or non-persistent, specified as boolean value ("true" or "false").

This will set the delivery mode accordingly, to either "PERSISTENT" or "NON-PERSISTENT".

Default it "true" aka delivery mode "PERSISTENT".

See also

`setDeliveryMode(int)` (p. 1212)

**6.198.3.32** `virtual void activemq::cmsutil::CmsTemplate::setExplicitQosEnabled ( bool explicitQosEnabled ) [inline, virtual]`

Set if the QOS values (deliveryMode, priority, timeToLive) should be used for sending a message.

See also

`setDeliveryMode` (p. 1212)

`setPriority` (p. 1213)

`setTimeToLive` (p. 1214)

**6.198.3.33** `virtual void activemq::cmsutil::CmsTemplate::setMessageIdEnabled ( bool messageIdEnabled ) [inline, virtual]`

**6.198.3.34** `virtual void activemq::cmsutil::CmsTemplate::setMessageTimestampEnabled ( bool messageTimestampEnabled ) [inline, virtual]`

**6.198.3.35** `virtual void activemq::cmsutil::CmsTemplate::setNoLocal ( bool noLocal ) [inline, virtual]`

**6.198.3.36** `virtual void activemq::cmsutil::CmsTemplate::setPriority ( int priority ) [inline, virtual]`

Set the priority of a message when sending.

Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

See also

`isExplicitQosEnabled` (p. 1208)

**6.198.3.37** `virtual void activemq::cmsutil::CmsTemplate::setPubSubDomain ( bool pubSubDomain ) [inline, virtual]`

Indicates whether the default destination is a topic (true) or a queue (false).

Calling this method will set the `defaultDestination` property to NULL.

#### Parameters

<i>pubSubDomain</i>	indicates whether to use pub-sub messaging (topics).
---------------------	--

Reimplemented from **activemq::cmsutil::CmsDestinationAccessor** (p. 1190).

References `activemq::cmsutil::CmsDestinationAccessor::setPubSubDomain()`.

**6.198.3.38** `virtual void activemq::cmsutil::CmsTemplate::setReceiveTimeout ( long long receiveTimeout ) [inline, virtual]`

**6.198.3.39** `virtual void activemq::cmsutil::CmsTemplate::setTimeToLive ( long long timeToLive ) [inline, virtual]`

Set the time-to-live of the message when sending.

Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

#### Parameters

<i>timeToLive</i>	the message's lifetime (in milliseconds)
-------------------	--

#### See also

**isExplicitQosEnabled** (p. 1208)

### 6.198.4 Friends And Related Function Documentation

6.198.4.1 friend class **ProducerExecutor** [friend]

6.198.4.2 friend class **ReceiveExecutor** [friend]

6.198.4.3 friend class **ResolveProducerExecutor** [friend]

6.198.4.4 friend class **ResolveReceiveExecutor** [friend]

6.198.4.5 friend class **SendExecutor** [friend]

### 6.198.5 Field Documentation

6.198.5.1 **const int activemq::cmsutil::CmsTemplate::DEFAULT\_PRIORITY**  
[static]

Default message priority.

6.198.5.2 **const long long activemq::cmsutil::CmsTemplate::DEFAULT\_TIME\_ -  
TO\_LIVE** [static]

My default, messages should live forever.

6.198.5.3 **const long long activemq::cmsutil::CmsTemplate::RECEIVE\_ -  
TIMEOUT\_INDEFINITE\_WAIT** [static]

Timeout value indicating a blocking receive without timeout.

6.198.5.4 **const long long activemq::cmsutil::CmsTemplate::RECEIVE\_ -  
TIMEOUT\_NO\_WAIT** [static]

Timeout value indicating that a receive operation should check if a message is immediately available without blocking.

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

## 6.199 code Struct Reference

```
#include <src/main/decaf/internal/util/zip/inftrees.h>
```

### Data Fields

- unsigned char **op**

- unsigned char **bits**
- unsigned short **val**

### 6.199.1 Field Documentation

#### 6.199.1.1 unsigned char code::bits

#### 6.199.1.2 unsigned char code::op

#### 6.199.1.3 unsigned short code::val

The documentation for this struct was generated from the following file:

- src/main/decaf/internal/util/zip/**inftrees.h**

## 6.200 decaf::util::Collection< E > Class Template Reference

The root interface in the collection hierarchy.

```
#include <src/main/decaf/util/Collection.h>
```

Inheritance diagram for decaf::util::Collection< E >:

### Public Member Functions

- virtual **~Collection** ()
- virtual bool **add** (const E &value)=0 throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )  
*Returns true if this collection changed as a result of the call.*
- virtual bool **addAll** (const **Collection**< E > &collection)=0 throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )  
*Adds all of the elements in the specified collection to this collection.*
- virtual void **clear** ()=0 throw ( lang::exceptions::UnsupportedOperationException )  
*Removes all of the elements from this collection (optional operation).*
- virtual bool **contains** (const E &value) const =0 throw ( lang::Exception )  
*Returns true if this collection contains the specified element.*



- virtual bool **containsAll** (const **Collection**< E > &collection) const =0 throw ( lang::Exception )  
*Returns true if this collection contains all of the elements in the specified collection.*
- virtual bool **equals** (const **Collection**< E > &value) const =0  
*Compares the passed collection to this one, if they contain the same elements, i.e.*
- virtual bool **isEmpty** () const =0
- virtual bool **remove** (const E &value)=0 throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )  
*Removes a single instance of the specified element from the collection.*
- virtual bool **removeAll** (const **Collection**< E > &collection)=0 throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )  
*Removes all this collection's elements that are also contained in the specified collection (optional operation).*
- virtual bool **retainAll** (const **Collection**< E > &collection)=0 throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )  
*Retains only the elements in this collection that are contained in the specified collection (optional operation).*
- virtual std::size\_t **size** () const =0  
*Returns the number of elements in this collection.*
- virtual std::vector< E > **toArray** () const =0  
*Returns an array containing all of the elements in this collection.*

### 6.200.1 Detailed Description

template<typename E> class decaf::util::Collection< E >

The root interface in the collection hierarchy. A collection represents a group of objects, known as its elements. Some collections allow duplicate elements and others do not. Some are ordered and others unordered. This interface is typically used to pass collections around and manipulate them where maximum generality is desired.

All general-purpose **Collection** (p. 1216) implementation classes (which typically implement **Collection** (p. 1216) indirectly through one of its subinterfaces) should provide two "standard" constructors: a void (no arguments) constructor, which creates an empty collection, and a constructor with a single argument of type **Collection** (p. 1216), which creates a new collection with the same elements as its argument. In effect, the latter constructor allows the user to copy any collection, producing an equivalent collection of the desired implementation type. There is no way to enforce this convention (as interfaces cannot contain constructors) but all of the general-purpose **Collection** (p. 1216) implementations in the Decaf platform libraries comply.

The "destructive" methods contained in this interface, that is, the methods that modify the collection on which they operate, are specified to throw `UnsupportedOperationException` if this collection does not support the operation. If this is the case, these methods may, but are not required to, throw an `UnsupportedOperationException` if the invocation would have no effect on the collection. For example, invoking the `addAll(Collection)` method on an unmodifiable collection may, but is not required to, throw the exception if the collection to be added is empty.

Many methods in Collections Framework interfaces are defined in terms of the `equals` method. For example, the specification for the `contains(Object o)` method says: "returns true if and only if this collection contains at least one element `e` such that `(o==null ? e==null : o.equals(e))`."

#### Since

1.0

### 6.200.2 Constructor & Destructor Documentation

**6.200.2.1** `template<typename E> virtual decaf::util::Collection< E >::~~Collection ( )`  
[inline, virtual]

### 6.200.3 Member Function Documentation

**6.200.3.1** `template<typename E> virtual bool decaf::util::Collection< E >::add ( const E & value ) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )` [pure virtual]

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 1216) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

#### Parameters

<i>value</i>	- reference to the element to add.
--------------	------------------------------------

**Returns**

true if the element was added

**Exceptions**

<i>UnsupportedOperationException</i>	
<i>IllegalArgumentException</i>	
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions

Implemented in **decaf::util::AbstractQueue< E >** (p. 177), **decaf::util::PriorityQueue< E >** (p. 3120), **decaf::util::StlList< E >** (p. 3700), **decaf::util::StlSet< E >** (p. 3734), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3700), **decaf::util::StlList< CompositeTask \* >** (p. 3700), **decaf::util::StlList< URI >** (p. 3700), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3700), **decaf::util::StlList< PrimitiveValueNode >** (p. 3700), **decaf::util::StlList< Pointer< Command > >** (p. 3700), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3700), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3700), **decaf::util::StlList< cms::Destination \* >** (p. 3700), **decaf::util::StlList< cms::Session \* >** (p. 3700), **decaf::util::StlList< cms::Connection \* >** (p. 3700), **decaf::util::StlSet< transport::TransportListener \* >** (p. 3734), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 3734), **decaf::util::StlSet< Resource \* >** (p. 3734), and **decaf::util::StlSet< ActiveMQSession \* >** (p. 3734).

```
6.200.3.2  template<typename E> virtual bool decaf::util::Collection<
           E >::addAll ( const Collection< E > & collection ) throw (
               lang::exceptions::UnsupportedOperationException,
               lang::exceptions::IllegalArgumentException,
               lang::exceptions::IllegalStateException ) [pure virtual]
```

Adds all of the elements in the specified collection to this collection.

The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

**Parameters**

<i>collection</i>	- <b>Collection</b> (p. 1216) whose elements are added to this one.
-------------------	---

**Returns**

true if this collection changed as a result of the call

**Exceptions**

<i>UnsupportedOperationException</i>	
<i>IllegalArgumentException</i>	

<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions
------------------------------	---

Implemented in `decaf::util::AbstractCollection< E >` (p. 163), `decaf::util::AbstractQueue< E >` (p. 177), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 163), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 163), `decaf::util::AbstractCollection< Resource * >` (p. 163), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 163), `decaf::util::AbstractCollection< CompositeTask * >` (p. 163), `decaf::util::AbstractCollection< URI >` (p. 163), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 163), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 163), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 163), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 163), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 163), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 163), `decaf::util::AbstractCollection< cms::Destination * >` (p. 163), `decaf::util::AbstractCollection< cms::Session * >` (p. 163), and `decaf::util::AbstractCollection< cms::Connection * >` (p. 163).

```
6.200.3.3  template<typename E> virtual void decaf::util::Collection< E >::clear ( )
            throw ( lang::exceptions::UnsupportedOperationException ) [pure
            virtual]
```

Removes all of the elements from this collection (optional operation).

This collection will be empty after this method returns unless it throws an exception.

### Exceptions

<i>UnsupportedOperationException</i>	
--------------------------------------	--

Implemented in `decaf::util::AbstractCollection< E >` (p. 163), `decaf::util::AbstractQueue< E >` (p. 178), `decaf::util::concurrent::SynchronousQueue< E >` (p. 3830), `decaf::util::PriorityQueue< E >` (p. 3120), `decaf::util::StlList< E >` (p. 3702), `decaf::util::StlSet< E >` (p. 3735), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 163), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 163), `decaf::util::AbstractCollection< Resource * >` (p. 163), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 163), `decaf::util::AbstractCollection< CompositeTask * >` (p. 163), `decaf::util::AbstractCollection< URI >` (p. 163), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 163), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 163), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 163), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 163), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 163), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 163), `decaf::util::AbstractCollection< cms::Destination * >` (p. 163), `decaf::util::AbstractCollection< cms::Session * >` (p. 163), `decaf::util::AbstractCollection< cms::Connection * >` (p. 163), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3702), `decaf::util::StlList< CompositeTask * >` (p. 3702), `decaf::util::StlList< URI >` (p. 3702), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3702), `decaf::util::StlList< PrimitiveValueNode >` (p. 3702), `decaf::util::StlList< Pointer< Command > >` (p. 3702), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3702), `decaf::util::StlList< cms::MessageProducer`

\* > (p. 3702), **decaf::util::StlList**< **cms::Destination** \* > (p. 3702), **decaf::util::StlList**< **cms::Session** \* > (p. 3702), **decaf::util::StlList**< **cms::Connection** \* > (p. 3702), **decaf::util::StlSet**< **transport::TransportListener** \* > (p. 3735), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 3735), **decaf::util::StlSet**< **Resource** \* > (p. 3735), and **decaf::util::StlSet**< **ActiveMQSession** \* > (p. 3735).

**6.200.3.4** `template<typename E> virtual bool decaf::util::Collection< E >::contains (const E & value ) const throw ( lang::Exception ) [pure virtual]`

Returns true if this collection contains the specified element.

More formally, returns true if and only if this collection contains at least one element *e* such that (*o*==null ? *e*==null : *o.equals(e)*).

#### Parameters

<i>value</i>	- value to check for presence in the collection
--------------	---

#### Returns

true if there is at least one of the elements in the collection

#### Exceptions

<i>Exception</i>	
------------------	--

Implemented in **decaf::util::AbstractCollection**< **E** > (p. 164), **decaf::util::StlList**< **E** > (p. 3702), **decaf::util::StlSet**< **E** > (p. 3736), **decaf::util::AbstractCollection**< **transport::TransportListener** \* > (p. 164), **decaf::util::AbstractCollection**< **Pointer**< **Synchronization** > > (p. 164), **decaf::util::AbstractCollection**< **Resource** \* > (p. 164), **decaf::util::AbstractCollection**< **cms::MessageConsumer** \* > (p. 164), **decaf::util::AbstractCollection**< **CompositeTask** \* > (p. 164), **decaf::util::AbstractCollection**< **URI** > (p. 164), **decaf::util::AbstractCollection**< **ActiveMQSession** \* > (p. 164), **decaf::util::AbstractCollection**< **Pointer**< **DestinationInfo** > > (p. 164), **decaf::util::AbstractCollection**< **PrimitiveValueNode** > (p. 164), **decaf::util::AbstractCollection**< **Pointer**< **Command** > > (p. 164), **decaf::util::AbstractCollection**< **Pointer**< **BackupTransport** > > (p. 164), **decaf::util::AbstractCollection**< **cms::MessageProducer** \* > (p. 164), **decaf::util::AbstractCollection**< **cms::Destination** \* > (p. 164), **decaf::util::AbstractCollection**< **cms::Session** \* > (p. 164), **decaf::util::AbstractCollection**< **cms::Connection** \* > (p. 164), **decaf::util::StlList**< **cms::MessageConsumer** \* > (p. 3702), **decaf::util::StlList**< **CompositeTask** \* > (p. 3702), **decaf::util::StlList**< **URI** > (p. 3702), **decaf::util::StlList**< **Pointer**< **DestinationInfo** > > (p. 3702), **decaf::util::StlList**< **PrimitiveValueNode** > (p. 3702), **decaf::util::StlList**< **Pointer**< **Command** > > (p. 3702), **decaf::util::StlList**< **Pointer**< **BackupTransport** > > (p. 3702), **decaf::util::StlList**< **cms::MessageProducer** \* > (p. 3702), **decaf::util::StlList**< **cms::Destination** \* > (p. 3702), **decaf::util::StlList**< **cms::Session** \* > (p. 3702), **decaf::util::StlList**< **cms::Connection** \* > (p. 3702), **decaf::util::StlSet**< **transport::TransportListener** \* > (p. 3736), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 3736), **decaf::util::StlSet**< **Resource** \* > (p. 3736), and **decaf::util::StlSet**< **ActiveMQSession** \* > (p. 3736).

```
6.200.3.5  template<typename E> virtual bool decaf::util::Collection< E >::containsAll (
            const Collection< E > & collection ) const throw ( lang::Exception )  [pure
            virtual]
```

Returns true if this collection contains all of the elements in the specified collection.

#### Parameters

<i>collection</i>	- <b>Collection</b> (p. 1216) to compare to this one.
-------------------	---

#### Exceptions

<i>Exception</i>
------------------

Implemented in **decaf::util::AbstractCollection< E >** (p. 165), **decaf::util::concurrent::SynchronousQueue< E >** (p. 3830), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 165), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 165), **decaf::util::AbstractCollection< Resource \* >** (p. 165), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 165), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 165), **decaf::util::AbstractCollection< URI >** (p. 165), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 165), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 165), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 165), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 165), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 165), **decaf::util::AbstractCollection< cms::MessageProducer \* >** (p. 165), **decaf::util::AbstractCollection< cms::Destination \* >** (p. 165), **decaf::util::AbstractCollection< cms::Session \* >** (p. 165), and **decaf::util::AbstractCollection< cms::Connection \* >** (p. 165).

```
6.200.3.6  template<typename E> virtual bool decaf::util::Collection< E >::equals (
            const Collection< E > & value ) const  [pure virtual]
```

Compares the passed collection to this one, if they contain the same elements, i.e. all their elements are equivalent, then it returns true.

#### Returns

true if the Collections contain the same elements.

Implemented in **decaf::util::AbstractCollection< E >** (p. 166), **decaf::util::concurrent::SynchronousQueue< E >** (p. 3832), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 166), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 166), **decaf::util::AbstractCollection< Resource \* >** (p. 166), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 166), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 166), **decaf::util::AbstractCollection< URI >** (p. 166), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 166), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 166), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 166), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 166), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 166), **decaf::util::AbstractCollection< cms::MessageProducer \* >** (p. 166), **decaf::util::AbstractCollection< cms::Destination**

\* > (p. 166), **decaf::util::AbstractCollection< cms::Session \* >** (p. 166), and **decaf::util::AbstractCollection< cms::Connection \* >** (p. 166).

**6.200.3.7** `template<typename E> virtual bool decaf::util::Collection< E >::isEmpty ( )`  
`const [pure virtual]`

### Returns

true if this collection contains no elements.

Implemented in **decaf::util::AbstractCollection< E >** (p. 166), **decaf::util::concurrent::SynchronousQueue< E >** (p. 3833), **decaf::util::StlList< E >** (p. 3704), **decaf::util::StlSet< E >** (p. 3736), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 166), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 166), **decaf::util::AbstractCollection< Resource \* >** (p. 166), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 166), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 166), **decaf::util::AbstractCollection< URI >** (p. 166), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 166), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 166), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 166), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 166), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 166), **decaf::util::AbstractCollection< cms::MessageProducer \* >** (p. 166), **decaf::util::AbstractCollection< cms::Destination \* >** (p. 166), **decaf::util::AbstractCollection< cms::Session \* >** (p. 166), **decaf::util::AbstractCollection< cms::Connection \* >** (p. 166), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3704), **decaf::util::StlList< CompositeTask \* >** (p. 3704), **decaf::util::StlList< URI >** (p. 3704), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3704), **decaf::util::StlList< PrimitiveValueNode >** (p. 3704), **decaf::util::StlList< Pointer< Command > >** (p. 3704), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3704), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3704), **decaf::util::StlList< cms::Destination \* >** (p. 3704), **decaf::util::StlList< cms::Session \* >** (p. 3704), **decaf::util::StlList< cms::Connection \* >** (p. 3704), **decaf::util::StlSet< transport::TransportListener \* >** (p. 3736), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 3736), **decaf::util::StlSet< Resource \* >** (p. 3736), and **decaf::util::StlSet< ActiveMQSession \* >** (p. 3736).

**6.200.3.8** `template<typename E> virtual bool decaf::util::Collection< E >::remove ( const E & value ) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException ) [pure virtual]`

Removes a single instance of the specified element from the collection.

More formally, removes an element *e* such that (*o*==null ? *e*==null : *o*.equals(*e*)), if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

### Parameters

<i>value</i>	- reference to the element to remove.
--------------	---------------------------------------

**Returns**

true if the collection was changed

**Exceptions**

<i>UnsupportedOperationException</i>	
<i>IllegalArgumentException</i>	

Implemented in `decaf::util::AbstractCollection< E >` (p. 168), `decaf::util::PriorityQueue< E >` (p. 3123), `decaf::util::StlList< E >` (p. 3706), `decaf::util::StlSet< E >` (p. 3737), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 168), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 168), `decaf::util::AbstractCollection< Resource * >` (p. 168), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 168), `decaf::util::AbstractCollection< CompositeTask * >` (p. 168), `decaf::util::AbstractCollection< URI >` (p. 168), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 168), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 168), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 168), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 168), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 168), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 168), `decaf::util::AbstractCollection< cms::Destination * >` (p. 168), `decaf::util::AbstractCollection< cms::Session * >` (p. 168), `decaf::util::AbstractCollection< cms::Connection * >` (p. 168), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3706), `decaf::util::StlList< CompositeTask * >` (p. 3706), `decaf::util::StlList< URI >` (p. 3706), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3706), `decaf::util::StlList< PrimitiveValueNode >` (p. 3706), `decaf::util::StlList< Pointer< Command > >` (p. 3706), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3706), `decaf::util::StlList< cms::MessageProducer * >` (p. 3706), `decaf::util::StlList< cms::Destination * >` (p. 3706), `decaf::util::StlList< cms::Session * >` (p. 3706), `decaf::util::StlList< cms::Connection * >` (p. 3706), `decaf::util::StlSet< transport::TransportListener * >` (p. 3737), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 3737), `decaf::util::StlSet< Resource * >` (p. 3737), and `decaf::util::StlSet< ActiveMQSession * >` (p. 3737).

```
6.200.3.9  template<typename E> virtual bool decaf::util::Collection<
            E >::removeAll( const Collection< E > & collection ) throw
            ( lang::exceptions::UnsupportedOperationException,
              lang::exceptions::IllegalArgumentException ) [pure virtual]
```

Removes all this collection's elements that are also contained in the specified collection (optional operation).

After this call returns, this collection will contain no elements in common with the specified collection.

**Parameters**

<i>collection</i>	- The <b>Collection</b> (p. 1216) whose elements are to be removed
-------------------	--



**Returns**

true if the collection changed as a result of this call

**Exceptions**

<i>UnsupportedOperationException</i>	
<i>IllegalArgumentException</i>	

Implemented in **decaf::util::AbstractCollection< E >** (p. 169), **decaf::util::AbstractSet< E >** (p. 181), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 169), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 169), **decaf::util::AbstractCollection< Resource \* >** (p. 169), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 169), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 169), **decaf::util::AbstractCollection< URI >** (p. 169), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 169), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 169), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 169), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 169), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 169), **decaf::util::AbstractCollection< cms::MessageProducer \* >** (p. 169), **decaf::util::AbstractCollection< cms::Destination \* >** (p. 169), **decaf::util::AbstractCollection< cms::Session \* >** (p. 169), **decaf::util::AbstractCollection< cms::Connection \* >** (p. 169), **decaf::util::AbstractSet< transport::TransportListener \* >** (p. 181), **decaf::util::AbstractSet< Pointer< Synchronization > >** (p. 181), **decaf::util::AbstractSet< Resource \* >** (p. 181), and **decaf::util::AbstractSet< ActiveMQSession \* >** (p. 181).

```
6.200.3.10  template<typename E> virtual bool decaf::util::Collection<
            E >::retainAll ( const Collection< E > & collection ) throw
            ( lang::exceptions::UnsupportedOperationException,
              lang::exceptions::IllegalArgumentException ) [pure virtual]
```

Retains only the elements in this collection that are contained in the specified collection (optional operation).

In other words, removes from this collection all of its elements that are not contained in the specified collection.

**Parameters**

<i>collection</i>	- The <b>Collection</b> (p. 1216) whose elements are to be retained
-------------------	---

**Returns**

true if the collection changed as a result of this call

**Exceptions**

<i>UnsupportedOperationException</i>	
--------------------------------------	--

<i>IllegalArgumentEx- ception</i>
---------------------------------------

Implemented in `decaf::util::AbstractCollection< E >` (p. 170), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 170), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 170), `decaf::util::AbstractCollection< Resource * >` (p. 170), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 170), `decaf::util::AbstractCollection< CompositeTask * >` (p. 170), `decaf::util::AbstractCollection< URI >` (p. 170), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 170), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 170), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 170), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 170), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 170), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 170), `decaf::util::AbstractCollection< cms::Destination * >` (p. 170), `decaf::util::AbstractCollection< cms::Session * >` (p. 170), and `decaf::util::AbstractCollection< cms::Connection * >` (p. 170).

**6.200.3.11** `template<typename E> virtual std::size_t decaf::util::Collection< E >::size ( ) const [pure virtual]`

Returns the number of elements in this collection.

If this collection contains more than `Integer.MAX_VALUE` elements, returns `Integer.MAX_VALUE`.

#### Returns

the number of elements in this collection

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3837), `decaf::util::PriorityQueue< E >` (p. 3124), `decaf::util::StlList< E >` (p. 3708), `decaf::util::StlSet< E >` (p. 3737), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3708), `decaf::util::StlList< CompositeTask * >` (p. 3708), `decaf::util::StlList< URI >` (p. 3708), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3708), `decaf::util::StlList< PrimitiveValueNode >` (p. 3708), `decaf::util::StlList< Pointer< Command > >` (p. 3708), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3708), `decaf::util::StlList< cms::MessageProducer * >` (p. 3708), `decaf::util::StlList< cms::Destination * >` (p. 3708), `decaf::util::StlList< cms::Session * >` (p. 3708), `decaf::util::StlList< cms::Connection * >` (p. 3708), `decaf::util::StlSet< transport::TransportListener * >` (p. 3737), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 3737), `decaf::util::StlSet< Resource * >` (p. 3737), and `decaf::util::StlSet< ActiveMQSession * >` (p. 3737).

Referenced by `decaf::util::AbstractCollection< cms::Connection * >::equals()`, `decaf::util::AbstractCollection< cms::Connection * >::isEmpty()`, `decaf::util::AbstractSet< ActiveMQSession * >::removeAll()`, and `decaf::util::AbstractCollection< cms::Connection * >::toArray()`.

**6.200.3.12** `template<typename E> virtual std::vector<E> decaf::util::Collection< E >::toArray ( ) const [pure virtual]`

Returns an array containing all of the elements in this collection.

If the collection makes any guarantees as to what order its elements are returned by its iterator, this method must return the elements in the same order.

This method acts as bridge between array-based and collection-based APIs.

### Returns

an array of the elements in this collection.

Implemented in `decaf::util::AbstractCollection< E >` (p. 170), `decaf::util::concurrent::SynchronousQueue< E >` (p. 3838), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 170), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 170), `decaf::util::AbstractCollection< Resource * >` (p. 170), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 170), `decaf::util::AbstractCollection< CompositeTask * >` (p. 170), `decaf::util::AbstractCollection< URI >` (p. 170), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 170), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 170), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 170), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 170), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 170), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 170), `decaf::util::AbstractCollection< cms::Destination * >` (p. 170), `decaf::util::AbstractCollection< cms::Session * >` (p. 170), and `decaf::util::AbstractCollection< cms::Connection * >` (p. 170).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Collection.h`

## 6.201 activemq::commands::Command Class Reference

```
#include <src/main/activemq/commands/Command.h>
```

Inheritance diagram for `activemq::commands::Command`:

### Public Member Functions

- virtual `~Command()`
- virtual void `setCommandId` (int id)=0  
*Sets the **Command** (p. 1227) Id of this **Message** (p. 2596).*
- virtual int `getCommandId` () const =0  
*Gets the **Command** (p. 1227) Id of this **Message** (p. 2596).*
- virtual void `setResponseRequired` (const bool required)=0  
*Set if this **Message** (p. 2596) requires a **Response** (p. 3379).*
- virtual bool `isResponseRequired` () const =0

Is a **Response** (p. 3379) required for this **Command** (p. 1227).

- virtual std::string **toString** () const =0

Returns a provider-specific string that provides information about the contents of the command.

- virtual **decaf::lang::Pointer< commands::Command > visit** (activemq::state::CommandVisitor \*visitor)=0 throw ( exceptions::ActiveMQException )

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

- virtual bool **isConnectionInfo** () const =0
- virtual bool **isConsumerInfo** () const =0
- virtual bool **isBrokerInfo** () const =0
- virtual bool **isKeepAliveInfo** () const =0
- virtual bool **isMessage** () const =0
- virtual bool **isMessageAck** () const =0
- virtual bool **isMessageDispatch** () const =0
- virtual bool **isMessageDispatchNotification** () const =0
- virtual bool **isProducerAck** () const =0
- virtual bool **isProducerInfo** () const =0
- virtual bool **isResponse** () const =0
- virtual bool **isRemoveInfo** () const =0
- virtual bool **isRemoveSubscriptionInfo** () const =0
- virtual bool **isShutdownInfo** () const =0
- virtual bool **isTransactionInfo** () const =0
- virtual bool **isWireFormatInfo** () const =0

### 6.201.1 Constructor & Destructor Documentation

- 6.201.1.1 virtual activemq::commands::Command::~~Command ( ) [inline, virtual]

### 6.201.2 Member Function Documentation

- 6.201.2.1 virtual int activemq::commands::Command::getCommandId ( ) const [pure virtual]

Gets the **Command** (p. 1227) Id of this **Message** (p. 2596).

#### Returns

**Command** (p. 1227) Id

Implemented in **activemq::commands::BaseCommand** (p. 767).

**6.201.2.2** `virtual bool activemq::commands::Command::isBrokerInfo ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 767), and **activemq::commands::BrokerInfo** (p. 907).

**6.201.2.3** `virtual bool activemq::commands::Command::isConnectionInfo ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 768), and **activemq::commands::ConnectionInfo** (p. 1397).

**6.201.2.4** `virtual bool activemq::commands::Command::isConsumerInfo ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 768), and **activemq::commands::ConsumerInfo** (p. 1506).

**6.201.2.5** `virtual bool activemq::commands::Command::isKeepAliveInfo ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 768), and **activemq::commands::KeepAliveInfo** (p. 2337).

**6.201.2.6** `virtual bool activemq::commands::Command::isMessage ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 768), and **activemq::commands::Message** (p. 2608).

**6.201.2.7** `virtual bool activemq::commands::Command::isMessageAck ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 768), and **activemq::commands::MessageAck** (p. 2646).

**6.201.2.8** `virtual bool activemq::commands::Command::isMessageDispatch ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 768), and **activemq::commands::MessageDispatch** (p. 2681).

**6.201.2.9** `virtual bool activemq::commands::Command::isMessageDispatchNotification ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 768), and **activemq::commands::MessageDispatch** (p. 2719).

**6.201.2.10** `virtual bool activemq::commands::Command::isProducerAck ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 769), and **activemq::commands::ProducerAck** (p. 3127).

**6.201.2.11** `virtual bool activemq::commands::Command::isProducerInfo ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 769), and **activemq::commands::ProducerInfo** (p. 3188).

**6.201.2.12** `virtual bool activemq::commands::Command::isRemoveInfo ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 769), and **activemq::commands::RemoveInfo** (p. 3286).

**6.201.2.13** `virtual bool activemq::commands::Command::isRemoveSubscriptionInfo ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 769), and **activemq::commands::RemoveSubscriptionInfo** (p. 3316).

**6.201.2.14** `virtual bool activemq::commands::Command::isResponse ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 769), and **activemq::commands::Response** (p. 3381).

**6.201.2.15** `virtual bool activemq::commands::Command::isResponseRequired ( ) const [pure virtual]`

Is a **Response** (p. 3379) required for this **Command** (p. 1227).

### Returns

true if a response is required.

Implemented in **activemq::commands::BaseCommand** (p. 769).

**6.201.2.16** `virtual bool activemq::commands::Command::isShutdownInfo ( ) const` `[pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 770), and **activemq::commands::ShutdownInfo** (p. 3575).

**6.201.2.17** `virtual bool activemq::commands::Command::isTransactionInfo ( ) const` `[pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 770), and **activemq::commands::TransactionInfo** (p. 3962).

**6.201.2.18** `virtual bool activemq::commands::Command::isWireFormatInfo ( ) const` `[pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 770), and **activemq::commands::WireFormatInfo** (p. 4101).

**6.201.2.19** `virtual void activemq::commands::Command::setCommandId ( int id )` `[pure virtual]`

Sets the **Command** (p. 1227) Id of this **Message** (p. 2596).

#### Parameters

<i>id</i>	<b>Command</b> (p. 1227) Id
-----------	-----------------------------

Implemented in **activemq::commands::BaseCommand** (p. 770).

**6.201.2.20** `virtual void activemq::commands::Command::setResponseRequired ( const bool required )` `[pure virtual]`

Set if this **Message** (p. 2596) requires a **Response** (p. 3379).

#### Parameters

<i>required</i>	true if response is required
-----------------	------------------------------

Implemented in **activemq::commands::BaseCommand** (p. 770).

**6.201.2.21** `virtual std::string activemq::commands::Command::toString ( ) const` `[pure virtual]`

Returns a provider-specific string that provides information about the contents of the command.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 841).

Implemented in `activemq::commands::ActiveMQBlobMessage` (p. 189), `activemq::commands::ActiveMQMapMessage` (p. 229), `activemq::commands::ActiveMQMapMessage` (p. 364), `activemq::commands::ActiveMQMessage` (p. 392), `activemq::commands::ActiveMQObjectMessage` (p. 440), `activemq::commands::ActiveMQStreamMessage` (p. 547), `activemq::commands::ActiveMQTextMessage` (p. 671), `activemq::commands::BaseCommand` (p. 770), `activemq::commands::BrokerInfo` (p. 908), `activemq::commands::ConnectionControl` (p. 1305), `activemq::commands::ConnectionError` (p. 1335), `activemq::commands::ConnectionInfo` (p. 1398), `activemq::commands::ConsumerControl` (p. 1445), `activemq::commands::ConsumerInfo` (p. 1507), `activemq::commands::ControlCommand` (p. 1538), `activemq::commands::DataArrayResponse` (p. 1574), `activemq::commands::DataResponse` (p. 1634), `activemq::commands::DestinationInfo` (p. 1783), `activemq::commands::ExceptionResponse` (p. 1897), `activemq::commands::FlushCommand` (p. 2001), `activemq::commands::IntegerResponse` (p. 2162), `activemq::commands::KeepAliveInfo` (p. 2337), `activemq::commands::Message` (p. 2611), `activemq::commands::MessageAck` (p. 2647), `activemq::commands::MessageDispatch` (p. 2682), `activemq::commands::MessageDispatchNotification` (p. 2719), `activemq::commands::MessagePull` (p. 2827), `activemq::commands::ProducerAck` (p. 3127), `activemq::commands::ProducerInfo` (p. 3189), `activemq::commands::RemoveInfo` (p. 3287), `activemq::commands::RemoveSubscriptionInfo` (p. 3317), `activemq::commands::ReplayCommand` (p. 3346), `activemq::commands::Response` (p. 3382), `activemq::commands::SessionInfo` (p. 3507), `activemq::commands::ShutdownInfo` (p. 3575), `activemq::commands::TransactionInfo` (p. 3963), and `activemq::commands::WireFormatInfo` (p. 4104).

```
6.201.2.22 virtual decaf::lang::Pointer<commands::Command>
activemq::commands::Command::visit ( activemq::state::CommandVisitor *
visitor ) throw ( exceptions::ActiveMQException ) [pure virtual]
```

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

### Returns

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implemented in `activemq::commands::BrokerError` (p. 873), `activemq::commands::BrokerInfo` (p. 909), `activemq::commands::ConnectionControl` (p. 1306), `activemq::commands::ConnectionError` (p. 1335), `activemq::commands::ConnectionInfo` (p. 1399), `activemq::commands::ConsumerControl` (p. 1446), `activemq::commands::ConsumerInfo` (p. 1508), `activemq::commands::ControlCommand` (p. 1539), `activemq::commands::DestinationInfo` (p. 1783), `activemq::commands::FlushCommand` (p. 2001), `activemq::commands::KeepAliveInfo` (p. 2338), `activemq::commands::Message` (p. 2612), `activemq::commands::MessageAck` (p. 2647), `activemq::commands::MessageDispatch` (p. 2682), `activemq::commands::MessageDispatchNotification` (p. 2720), `activemq::commands::MessagePull` (p. 2828), `activemq::commands::ProducerAck` (p. 3128), `activemq::commands::ProducerInfo` (p. 3189), `activemq::commands::RemoveInfo` (p. 3287), `activemq::commands::RemoveSubscriptionInfo` (p. 3317), `activemq::commands::ReplayCommand` (p. 3346), `activemq::commands::Response` (p. 3382), `activemq::commands::SessionInfo` (p. 3508), `activemq::commands::ShutdownInfo` (p. 3575), `activemq::commands::TransactionInfo` (p. 3963), and `activemq::commands::WireFormatInfo` (p. 4104).

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/Command.h`



## 6.202 activemq::state::CommandVisitor Class Reference

Interface for an Object that can visit the various Command Objects that are sent from and to this client.

```
#include <src/main/activemq/state/CommandVisitor.h>
```

Inheritance diagram for activemq::state::CommandVisitor:

### Public Member Functions

- virtual `~CommandVisitor()`
- virtual `decaf::lang::Pointer< commands::Command > processTransactionInfo (commands::TransactionInfo *info)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processRemoveInfo (commands::RemoveInfo *info)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processConnectionInfo (commands::ConnectionInfo *info)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processSessionInfo (commands::SessionInfo *info)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processProducerInfo (commands::ProducerInfo *info)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processConsumerInfo (commands::ConsumerInfo *info)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processRemoveConnection (commands::ConnectionId *id)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processRemoveSession (commands::SessionId *id)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processRemoveProducer (commands::ProducerId *id)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processRemoveConsumer (commands::ConsumerId *id)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processDestinationInfo (commands::DestinationInfo *info)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processRemoveDestination (commands::DestinationInfo *info)=0` throw ( exceptions::ActiveMQException )
- virtual `decaf::lang::Pointer< commands::Command > processRemoveSubscriptionInfo (commands::RemoveSubscriptionInfo *info)=0` throw ( exceptions::ActiveMQException )

- virtual **decaf::lang::Pointer**< **commands::Command** > **processMessage** (**commands::Message** \*send)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processMessageAck** (**commands::MessageAck** \*ack)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processMessagePull** (**commands::MessagePull** \*pull)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processBeginTransaction** (**commands::TransactionInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processPrepareTransaction** (**commands::TransactionInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processCommitTransactionOnePhase** (**commands::TransactionInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processCommitTransactionTwoPhase** (**commands::TransactionInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processRollbackTransaction** (**commands::TransactionInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processWireFormat** (**commands::WireFormatInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processKeepAliveInfo** (**commands::KeepAliveInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processShutdownInfo** (**commands::ShutdownInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processFlushCommand** (**commands::FlushCommand** \*command)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processBrokerInfo** (**commands::BrokerInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processRecoverTransactions** (**commands::TransactionInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processForgetTransaction** (**commands::TransactionInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processEndTransaction** (**commands::TransactionInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processMessageDispatchNotification** (**commands::MessageDispatchNotification** \*notification)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processProducerAck** (**commands::ProducerAck** \*ack)=0 throw ( exceptions::ActiveMQException )

- virtual **decaf::lang::Pointer< commands::Command > processMessageDispatch** (commands::MessageDispatch \*dispatch)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processControlCommand** (commands::ControlCommand \*command)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processConnectionError** (commands::ConnectionError \*error)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processConnectionControl** (commands::ConnectionControl \*control)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processConsumerControl** (commands::ConsumerControl \*control)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processBrokerError** (commands::BrokerError \*error)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processReplayCommand** (commands::ReplayCommand \*replay)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processResponse** (commands::Response \*response)=0 throw ( exceptions::ActiveMQException )

### 6.202.1 Detailed Description

Interface for an Object that can visit the various Command Objects that are sent from and to this client. The Commands themselves implement a `visit` method that is called with an instance of this interface and each one then call the appropriate `processXXX` method.

Since

3.0

### 6.202.2 Constructor & Destructor Documentation

6.202.2.1 virtual **activemq::state::CommandVisitor::~~CommandVisitor** ( ) [inline, virtual]

### 6.202.3 Member Function Documentation

6.202.3.1 virtual **decaf::lang::Pointer< commands::Command > activemq::state::CommandVisitor::processBeginTransaction** ( commands::TransactionInfo \* *info* ) throw ( exceptions::ActiveMQException ) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 1435).

- 6.202.3.2 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitor::processBrokerError`  
`( commands::BrokerError * error ) throw (`  
`exceptions::ActiveMQException) [pure virtual]`
- 6.202.3.3 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitor::processBrokerInfo ( commands::BrokerInfo *  
info ) throw ( exceptions::ActiveMQException) [pure virtual]`
- 6.202.3.4 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitor::processCommitTransactionOnePhase`  
`( commands::TransactionInfo * info ) throw (`  
`exceptions::ActiveMQException) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1436).

- 6.202.3.5 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitor::processCommitTransactionTwoPhase`  
`( commands::TransactionInfo * info ) throw (`  
`exceptions::ActiveMQException) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1436).

- 6.202.3.6 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitor::processConnectionControl`  
`( commands::ConnectionControl * control ) throw (`  
`exceptions::ActiveMQException) [pure virtual]`
- 6.202.3.7 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitor::processConnectionError`  
`( commands::ConnectionError * error ) throw (`  
`exceptions::ActiveMQException) [pure virtual]`
- 6.202.3.8 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitor::processConnectionInfo`  
`( commands::ConnectionInfo * info ) throw (`  
`exceptions::ActiveMQException) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1436).

6.202.3.9 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processConsumerControl  
( commands::ConsumerControl \* *control* ) throw (   
exceptions::ActiveMQException ) [pure virtual]

6.202.3.10 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processConsumerInfo  
( commands::ConsumerInfo \* *info* ) throw (   
exceptions::ActiveMQException ) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1436).

6.202.3.11 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processControlCommand  
( commands::ControlCommand \* *command* ) throw (   
exceptions::ActiveMQException ) [pure virtual]

6.202.3.12 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processDestinationInfo  
( commands::DestinationInfo \* *info* ) throw (   
exceptions::ActiveMQException ) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1436).

6.202.3.13 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processEndTransaction  
( commands::TransactionInfo \* *info* ) throw (   
exceptions::ActiveMQException ) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1436).

- 6.202.3.14 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitor::processFlushCommand`  
`( commands::FlushCommand * command ) throw (`  
`exceptions::ActiveMQException ) [pure virtual]`
- 6.202.3.15 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitor::processForgetTransaction`  
`( commands::TransactionInfo * info ) throw (`  
`exceptions::ActiveMQException ) [pure virtual]`
- 6.202.3.16 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitor::processKeepAliveInfo`  
`( commands::KeepAliveInfo * info ) throw (`  
`exceptions::ActiveMQException ) [pure virtual]`
- 6.202.3.17 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitor::processMessage ( commands::Message *`  
`send ) throw ( exceptions::ActiveMQException ) [pure virtual]`
- Implemented in `activemq::state::ConnectionStateTracker` (p. 1437).
- 6.202.3.18 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitor::processMessageAck (`  
`commands::MessageAck * ack ) throw ( exceptions::ActiveMQException`  
`) [pure virtual]`
- Implemented in `activemq::state::ConnectionStateTracker` (p. 1437).

- 6.202.3.19 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processMessageDispatch  
( commands::MessageDispatch * dispatch ) throw ( exceptions::ActiveMQException ) [pure virtual]`
- 6.202.3.20 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processMessageDispatchNotification ( commands::MessageDispatchNotification * notification ) throw ( exceptions::ActiveMQException ) [pure virtual]`
- 6.202.3.21 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processMessagePull ( commands::MessagePull * pull ) throw ( exceptions::ActiveMQException ) [pure virtual]`
- 6.202.3.22 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processPrepareTransaction  
( commands::TransactionInfo * info ) throw ( exceptions::ActiveMQException ) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1437).

- 6.202.3.23 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processProducerAck  
( commands::ProducerAck * ack ) throw ( exceptions::ActiveMQException ) [pure virtual]`
- 6.202.3.24 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processProducerInfo  
( commands::ProducerInfo * info ) throw ( exceptions::ActiveMQException ) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1437).

- 6.202.3.25 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRecoverTransactions  
( commands::TransactionInfo * info ) throw ( exceptions::ActiveMQException ) [pure virtual]`
- 6.202.3.26 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRemoveConnection ( commands::ConnectionId * id ) throw ( exceptions::ActiveMQException ) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1437).

**6.202.3.27** `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRemoveConsumer (  
commands::ConsumerId * id ) throw ( exceptions::ActiveMQException )  
[pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1437).

**6.202.3.28** `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRemoveDestination  
( commands::DestinationInfo * info ) throw (   
exceptions::ActiveMQException ) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1438).

**6.202.3.29** `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRemoveInfo (   
commands::RemoveInfo * info ) throw ( exceptions::ActiveMQException  
) [pure virtual]`

Implemented in `activemq::state::CommandVisitorAdapter` (p. 1245).

**6.202.3.30** `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRemoveProducer (   
commands::ProducerId * id ) throw ( exceptions::ActiveMQException )  
[pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1438).

**6.202.3.31** `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRemoveSession (   
commands::SessionId * id ) throw ( exceptions::ActiveMQException )  
[pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1438).



- 6.202.3.32 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRemoveSubscriptionInfo  
( commands::RemoveSubscriptionInfo * info ) throw ( exceptions::ActiveMQException ) [pure virtual]`
- 6.202.3.33 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processReplayCommand  
( commands::ReplayCommand * replay ) throw ( exceptions::ActiveMQException ) [pure virtual]`
- 6.202.3.34 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processResponse ( commands::Response  
* response ) throw ( exceptions::ActiveMQException ) [pure  
virtual]`
- 6.202.3.35 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRollbackTransaction  
( commands::TransactionInfo * info ) throw ( exceptions::ActiveMQException ) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1438).

- 6.202.3.36 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processSessionInfo ( commands::SessionInfo * info ) throw ( exceptions::ActiveMQException )  
[pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1438).

- 6.202.3.37 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processShutdownInfo  
( commands::ShutdownInfo * info ) throw ( exceptions::ActiveMQException ) [pure virtual]`
- 6.202.3.38 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processTransactionInfo  
( commands::TransactionInfo * info ) throw ( exceptions::ActiveMQException ) [pure virtual]`

Implemented in `activemq::state::CommandVisitorAdapter` (p. 1247).

- 6.202.3.39 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processWireFormat  
( commands::WireFormatInfo * info ) throw ( exceptions::ActiveMQException ) [pure virtual]`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/CommandVisitor.h`

## 6.203 `activemq::state::CommandVisitorAdapter` Class Reference

Default Implementation of a **CommandVisitor** (p. 1233) that returns NULL for all calls.

```
#include <src/main/activemq/state/CommandVisitorAdapter.h>
```

Inheritance diagram for `activemq::state::CommandVisitorAdapter`:

### Public Member Functions

- `virtual ~CommandVisitorAdapter ()`
- `virtual decaf::lang::Pointer< commands::Command > processRemoveConnection (commands::ConnectionId *id AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- `virtual decaf::lang::Pointer< commands::Command > processRemoveSession (commands::SessionId *id AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- `virtual decaf::lang::Pointer< commands::Command > processRemoveProducer (commands::ProducerId *id AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- `virtual decaf::lang::Pointer< commands::Command > processRemoveConsumer (commands::ConsumerId *id AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- `virtual decaf::lang::Pointer< commands::Command > processDestinationInfo (commands::DestinationInfo *info AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- `virtual decaf::lang::Pointer< commands::Command > processRemoveDestination (commands::DestinationInfo *info AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- `virtual decaf::lang::Pointer< commands::Command > processRemoveSubscriptionInfo (commands::RemoveSubscriptionInfo *info AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- `virtual decaf::lang::Pointer< commands::Command > processMessage (commands::Message *send AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- `virtual decaf::lang::Pointer< commands::Command > processMessageAck (commands::MessageAck *ack AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- `virtual decaf::lang::Pointer< commands::Command > processMessagePull (commands::MessagePull *pull AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`

- virtual **decaf::lang::Pointer< commands::Command > processBeginTransaction** (**commands::TransactionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processPrepareTransaction** (**commands::TransactionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processCommitTransactionOnePhase** (**commands::TransactionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processCommitTransactionTwoPhase** (**commands::TransactionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processRollbackTransaction** (**commands::TransactionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processWireFormat** (**commands::WireFormatInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processKeepAliveInfo** (**commands::KeepAliveInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processShutdownInfo** (**commands::ShutdownInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processFlushCommand** (**commands::FlushCommand** \*command AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processBrokerInfo** (**commands::BrokerInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processRecoverTransactions** (**commands::TransactionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processForgetTransaction** (**commands::TransactionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processEndTransaction** (**commands::TransactionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processMessageDispatchNotification** (**commands::MessageDispatchNotification** \*notification AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processProducerAck** (**commands::ProducerAck** \*ack AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processMessageDispatch** (**commands::MessageDispatch** \*dispatch AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )

- virtual **decaf::lang::Pointer**< **commands::Command** > **processControlCommand** (**commands::ControlCommand** \*command AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConnectionError** (**commands::ConnectionError** \*error AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConnectionControl** (**commands::ConnectionControl** \*control AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConsumerControl** (**commands::ConsumerControl** \*control AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processBrokerError** (**commands::BrokerError** \*error AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processReplayCommand** (**commands::ReplayCommand** \*replay AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processResponse** (**commands::Response** \*response AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConnectionInfo** (**commands::ConnectionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processSessionInfo** (**commands::SessionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processProducerInfo** (**commands::ProducerInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConsumerInfo** (**commands::ConsumerInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processTransactionInfo** (**commands::TransactionInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processRemoveInfo** (**commands::RemoveInfo** \*info) throw ( exceptions::ActiveMQException )

### 6.203.1 Detailed Description

Default Implementation of a **CommandVisitor** (p. 1233) that returns NULL for all calls.

Since

3.0

## 6.203.2 Constructor & Destructor Documentation

6.203.2.1 virtual activemq::state::CommandVisitorAdapter::~~CommandVisitorAdapter ( )  
[inline, virtual]

## 6.203.3 Member Function Documentation

6.203.3.1 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processBeginTransaction (   
commands::TransactionInfo \*info *AMQCPP\_UNUSED* ) throw (   
exceptions::ActiveMQException ) [inline, virtual]

6.203.3.2 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processBrokerError (   
commands::BrokerError \*error *AMQCPP\_UNUSED* ) throw (   
exceptions::ActiveMQException ) [inline, virtual]

6.203.3.3 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processBrokerInfo (   
commands::BrokerInfo \*info *AMQCPP\_UNUSED* ) throw (   
exceptions::ActiveMQException ) [inline, virtual]

6.203.3.4 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processCommitTransactionOnePhase  
( commands::TransactionInfo \*info *AMQCPP\_UNUSED* ) throw (   
exceptions::ActiveMQException ) [inline, virtual]

6.203.3.5 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processCommitTransactionTwoPhase  
( commands::TransactionInfo \*info *AMQCPP\_UNUSED* ) throw (   
exceptions::ActiveMQException ) [inline, virtual]

6.203.3.6 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processConnectionControl (   
commands::ConnectionControl \*control *AMQCPP\_UNUSED* ) throw (   
exceptions::ActiveMQException ) [inline, virtual]

6.203.3.7 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processConnectionError (   
commands::ConnectionError \*error *AMQCPP\_UNUSED* ) throw (   
exceptions::ActiveMQException ) [inline, virtual]

6.203.3.8 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processConnectionInfo (   
commands::ConnectionInfo \*info *AMQCPP\_UNUSED* ) throw (   
exceptions::ActiveMQException ) [inline, virtual]

6.203.3.9 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processConsumerControl (   
commands::ConsumerControl \*control *AMQCPP\_UNUSED* ) throw (   
exceptions::ActiveMQException ) [inline, virtual]

6.203.3.10 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processConsumerInfo (   
commands::ConsumerInfo \*info *AMQCPP\_UNUSED* ) throw (   
exceptions::ActiveMQException ) [inline, virtual]

6.203.3.11 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processControlCommand (

References `activemq::commands::ConnectionId::ID_CONNECTIONID`, `activemq::commands::ConsumerId::ID_CONSUMERID`, `activemq::commands::ProducerId::ID_PRODUCERID`, and `activemq::commands::SessionId::ID_SESSIONID`.

- 6.203.3.30 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processRemoveProducer  
( commands::ProducerId *id AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.31 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processRemoveSession  
( commands::SessionId *id AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.32 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processRemoveSubscriptionInfo (  
commands::RemoveSubscriptionInfo *info AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.33 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processReplayCommand (  
commands::ReplayCommand *replay AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.34 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processResponse (  
commands::Response *response AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.35 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processRollbackTransaction  
( commands::TransactionInfo *info AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.36 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processSessionInfo  
( commands::SessionInfo *info AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.37 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processShutdownInfo (  
commands::ShutdownInfo *info AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.38 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processTransactionInfo  
( commands::TransactionInfo * info ) throw (  
exceptions::ActiveMQException) [inline, virtual]`

Implements `activemq::state::CommandVisitor` (p. 1241).

References `activemq::core::ActiveMQConstants::TRANSACTION_STATE_BEGIN`, `activemq::core::ActiveMQConstants::TRANSACTION_STATE_COMMITONEPHASE`, `activemq::core::ActiveMQConstants::TRANSACTION_STATE_COMMITTWO PHASE`, `activemq::core::ActiveMQConstants::TRANSACTION_`

STATE\_END, activemq::core::ActiveMQConstants::TRANSACTION\_STATE\_FORGET, activemq::core::ActiveMQConstants::TRANSACTION\_STATE\_PREPARE, activemq::core::ActiveMQConstants::TRANSACTION\_STATE\_RECOVER, and activemq::core::ActiveMQConstants::TRANSACTION\_STATE\_ROLLBACK.

```
6.203.3.39 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processWireFormat (
    commands::WireFormatInfo *info AMQCPP_UNUSED ) throw (
    exceptions::ActiveMQException ) [inline, virtual]
```

The documentation for this class was generated from the following file:

- src/main/activemq/state/CommandVisitorAdapter.h

## 6.204 decaf::lang::Comparable< T > Class Template Reference

This interface imposes a total ordering on the objects of each class that implements it.

```
#include <src/main/decaf/lang/Comparable.h>
```

### Public Member Functions

- virtual **~Comparable** ()
- virtual int **compareTo** (const T &value) const =0  
*Compares this object with the specified object for order.*
- virtual bool **equals** (const T &value) const =0
- virtual bool **operator==** (const T &value) const =0  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const T &value) const =0  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*

### 6.204.1 Detailed Description

```
template<typename T> class decaf::lang::Comparable< T >
```

This interface imposes a total ordering on the objects of each class that implements it. This ordering is referred to as the class's natural ordering, and the class's compareTo method is referred to as its natural comparison method.



## 6.204.2 Constructor & Destructor Documentation

6.204.2.1 `template<typename T> virtual decaf::lang::Comparable< T >::~~Comparable( ) [inline, virtual]`

## 6.204.3 Member Function Documentation

6.204.3.1 `template<typename T> virtual int decaf::lang::Comparable< T >::compareTo ( const T & value ) const [pure virtual]`

Compares this object with the specified object for order.

Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

In the foregoing description, the notation `sgn(expression)` designates the mathematical signum function, which is defined to return one of -1, 0, or 1 according to whether the value of expression is negative, zero or positive. The implementor must ensure `sgn(x.compareTo(y)) == -sgn(y.compareTo(x))` for all x and y. (This implies that `x.compareTo(y)` must throw an exception iff `y.compareTo(x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `(x.compareTo(y)>0 && y.compareTo(z)>0)` implies `x.compareTo(z)>0`.

Finally, the implementer must ensure that `x.compareTo(y)==0` implies that `sgn(x.compareTo(z)) == sgn(y.compareTo(z))`, for all z.

It is strongly recommended, but not strictly required that `(x.compareTo(y)==0) == (x.equals(y))`. Generally speaking, any class that implements the **Comparable** (p. 1248) interface and violates this condition should clearly indicate this fact. The recommended language is "Note: this class has a natural ordering that is inconsistent with equals."

### Parameters

<i>value</i>	- the Object to be compared.
--------------	------------------------------

### Returns

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Implemented in **decaf::lang::Boolean** (p. 857), **decaf::lang::Boolean** (p. 858), **decaf::lang::Byte** (p. 972), **decaf::lang::Byte** (p. 972), **decaf::lang::Character** (p. 1129), **decaf::lang::Character** (p. 1129), **decaf::lang::Double** (p. 1845), **decaf::lang::Double** (p. 1845), **decaf::lang::Float** (p. 1965), **decaf::lang::Float** (p. 1965), **decaf::lang::Integer** (p. 2148), **decaf::lang::Integer** (p. 2147), **decaf::lang::Long** (p. 2499), **decaf::lang::Long** (p. 2499), **decaf::lang::Short** (p. 3542), and **decaf::lang::Short** (p. 3541).

**6.204.3.2** `template<typename T> virtual bool decaf::lang::Comparable< T >::equals ( const T & value ) const` [pure virtual]

#### Returns

true if this value is considered equal to the passed value.

Implemented in **decaf::lang::Boolean** (p. 858), **decaf::lang::Boolean** (p. 858), **decaf::lang::Byte** (p. 974), **decaf::lang::Byte** (p. 974), **decaf::lang::Character** (p. 1130), **decaf::lang::Character** (p. 1130), **decaf::lang::Double** (p. 1847), **decaf::lang::Double** (p. 1847), **decaf::lang::Float** (p. 1966), **decaf::lang::Float** (p. 1966), **decaf::lang::Integer** (p. 2149), **decaf::lang::Integer** (p. 2149), **decaf::lang::Long** (p. 2500), **decaf::lang::Long** (p. 2501), **decaf::lang::Short** (p. 3543), and **decaf::lang::Short** (p. 3543).

**6.204.3.3** `template<typename T> virtual bool decaf::lang::Comparable< T >::operator< ( const T & value ) const` [pure virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

#### Returns

true if this object is equal to the one passed.

Implemented in **decaf::lang::Boolean** (p. 859), **decaf::lang::Boolean** (p. 858), **decaf::lang::Byte** (p. 975), **decaf::lang::Byte** (p. 975), **decaf::lang::Character** (p. 1132), **decaf::lang::Character** (p. 1133), **decaf::lang::Double** (p. 1849), **decaf::lang::Double** (p. 1849), **decaf::lang::Float** (p. 1969), **decaf::lang::Float** (p. 1969), **decaf::lang::Integer** (p. 2152), **decaf::lang::Integer** (p. 2151), **decaf::lang::Long** (p. 2503), **decaf::lang::Long** (p. 2503), **decaf::lang::Short** (p. 3544), and **decaf::lang::Short** (p. 3544).

**6.204.3.4** `template<typename T> virtual bool decaf::lang::Comparable< T >::operator== ( const T & value ) const` [pure virtual]

Compares equality between this object and the one passed.

#### Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

#### Returns

true if this object is equal to the one passed.

Implemented in **decaf::lang::Boolean** (p. 859), **decaf::lang::Boolean** (p. 859), **decaf::lang::Byte** (p. 975), **decaf::lang::Byte** (p. 976), **decaf::lang::Character** (p. 1133), **decaf::lang::Character** (p. 1133), **decaf::lang::Double** (p. 1849), **decaf::lang::Double** (p. 1850), **decaf::lang::Float** (p. 1970), **decaf::lang::Float** (p. 1970), **decaf::lang::Integer** (p. 2152), **decaf::lang::Integer** (p. 2152), **decaf::lang::Long** (p. 2504), **decaf::lang::Long** (p. 2504), **decaf::lang::Short** (p. 3544), and **decaf::lang::Short** (p. 3545).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/Comparable.h

## 6.205 decaf::util::Comparator< T > Class Template Reference

A comparison function, which imposes a total ordering on some collection of objects.

```
#include <src/main/decaf/util/Comparator.h>
```

Inheritance diagram for decaf::util::Comparator< T >:

### Public Member Functions

- virtual **~Comparator** ()
- virtual bool **operator**() (const T &left, const T &right) const =0

*Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 1251) to be passed to an STL **Map** (p. 2538) for use as the sorting criteria.*

- virtual int **compare** (const T &o1, const T &o2) const =0

*Compares its two arguments for order.*

### 6.205.1 Detailed Description

```
template<typename T> class decaf::util::Comparator< T >
```

A comparison function, which imposes a total ordering on some collection of objects. Comparators can be passed to a sort method (such as Collections.sort) to allow precise control over the sort order. Comparators can also be used to control the order of certain data structures.

The ordering imposed by a **Comparator** (p. 1251) c on a set of elements S is said to be consistent with equals if and only if ( compare( e1, e2 ) == 0 ) has the same boolean value as ( e1 == e2 ) for every e1 and e2 in S.

## 6.205.2 Constructor & Destructor Documentation

6.205.2.1 `template<typename T> virtual decaf::util::Comparator< T >::~Comparator( ) [inline, virtual]`

## 6.205.3 Member Function Documentation

6.205.3.1 `template<typename T> virtual int decaf::util::Comparator< T >::compare ( const T & o1, const T & o2 ) const [pure virtual]`

Compares its two arguments for order.

Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

The implementor must ensure that `sgn( compare(x, y)) == -sgn(compare(y, x) )` for all `x` and `y`. (This implies that `compare(x, y)` must throw an exception if and only if `compare(y, x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `((compare(x, y)>0) && (compare(y, z)>0))` implies `compare(x, z)>0`.

Finally, the implementer must ensure that `compare(x, y)==0` implies that `sgn(compare(x, z))==sgn(compare(y, z))` for all `z`.

It is generally the case, but not strictly required that `(compare(x, y)==0) == ( x == y )`. Generally speaking, any comparator that violates this condition should clearly indicate this fact. The recommended language is "Note: this comparator imposes orderings that are inconsistent with equals."

### Parameters

<i>o1</i>	- the first object to be compared
<i>o2</i>	- the second object to be compared

### Returns

a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

Implemented in `decaf::util::comparators::Less< E >` (p. 2400).

6.205.3.2 `template<typename T> virtual bool decaf::util::Comparator< T >::operator() ( const T & left, const T & right ) const [pure virtual]`

Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 1251) to be passed to an STL **Map** (p. 2538) for use as the sorting criteria.

### Parameters

<i>left</i>	- the Left hand side operand.
<i>right</i>	- the Right hand side operand.

**Returns**

true if the vale of left is less than the value of right.

Implemented in **decaf::util::comparators::Less< E >** (p. 2401).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Comparator.h`

## 6.206 activemq::util::CompositeData Class Reference

Represents a Composite URI.

```
#include <src/main/activemq/util/CompositeData.h>
```

**Public Member Functions**

- **CompositeData** ()
- virtual **~CompositeData** ()
- **StlList< URI > & getComponents** ()
- const **StlList< URI > & getComponents** () const
- void **setComponents** (const **StlList< URI > &components**)
- std::string **getFragment** () const
- void **setFragment** (const std::string &fragment)
- const **Properties & getParameters** () const
- void **setParameters** (const **Properties &parameters**)
- std::string **getScheme** () const
- void **setScheme** (const std::string &scheme)
- std::string **getPath** () const
- void **setPath** (const std::string &path)
- std::string **getHost** () const
- void **setHost** (const std::string &host)
- **URI toURI** () const throw ( decaf::net::URISyntaxException )

### 6.206.1 Detailed Description

Represents a Composite URI.

**Since**

3.0

## 6.206.2 Constructor & Destructor Documentation

6.206.2.1 `activemq::util::CompositeData::CompositeData ( )`

6.206.2.2 `virtual activemq::util::CompositeData::~~CompositeData ( )` `[virtual]`

## 6.206.3 Member Function Documentation

6.206.3.1 `StlList<URI>& activemq::util::CompositeData::getComponents ( )`  
`[inline]`

6.206.3.2 `const StlList<URI>& activemq::util::CompositeData::getComponents ( ) const`  
`[inline]`

6.206.3.3 `std::string activemq::util::CompositeData::getFragment ( ) const` `[inline]`

6.206.3.4 `std::string activemq::util::CompositeData::getHost ( ) const` `[inline]`

6.206.3.5 `const Properties& activemq::util::CompositeData::getParameters ( ) const`  
`[inline]`

6.206.3.6 `std::string activemq::util::CompositeData::getPath ( ) const` `[inline]`

6.206.3.7 `std::string activemq::util::CompositeData::getScheme ( ) const` `[inline]`

6.206.3.8 `void activemq::util::CompositeData::setComponents ( const StlList< URI > & components )` `[inline]`

6.206.3.9 `void activemq::util::CompositeData::setFragment ( const std::string & fragment )`  
`[inline]`

6.206.3.10 `void activemq::util::CompositeData::setHost ( const std::string & host )`  
`[inline]`

6.206.3.11 `void activemq::util::CompositeData::setParameters ( const Properties & parameters )` `[inline]`

6.206.3.12 `void activemq::util::CompositeData::setPath ( const std::string & path )`  
`[inline]`

6.206.3.13 `void activemq::util::CompositeData::setScheme ( const std::string & scheme )`  
`[inline]`

6.206.3.14 `URI activemq::util::CompositeData::toURI ( ) const throw ( decaf::net::URISyntaxException )`

The documentation for this class was generated from the following file:

- `src/main/activemq/util/CompositeData.h`

## 6.207 activemq::threads::CompositeTask Class Reference

Represents a single task that can be part of a set of Tasks that are contained in a **CompositeTaskRunner** (p.1256).

```
#include <src/main/activemq/threads/CompositeTask.h>
```

Inheritance diagram for activemq::threads::CompositeTask:

### Public Member Functions

- virtual **~CompositeTask** ()
- virtual bool **isPending** () const =0

*Indicates whether this task has any pending work that needs to be done, if not then it is skipped and the next **Task** (p.3846) in the CompositeTaskRunner's list of tasks is checked, if none of the tasks have any pending work to do, then the runner can go to sleep until it awakened by a call to wakeup.*

### 6.207.1 Detailed Description

Represents a single task that can be part of a set of Tasks that are contained in a **CompositeTaskRunner** (p.1256).

Since

3.0

### 6.207.2 Constructor & Destructor Documentation

6.207.2.1 **virtual activemq::threads::CompositeTask::~CompositeTask** ( ) [inline, virtual]

### 6.207.3 Member Function Documentation

6.207.3.1 **virtual bool activemq::threads::CompositeTask::isPending** ( ) const [pure virtual]

Indicates whether this task has any pending work that needs to be done, if not then it is skipped and the next **Task** (p.3846) in the CompositeTaskRunner's list of tasks is checked, if none of the tasks have any pending work to do, then the runner can go to sleep until it awakened by a call to wakeup.

Since

3.0

Implemented in `activemq::transport::failover::BackupTransportPool` (p. 764), `activemq::transport::failover::CloseTransportsTask` (p. 1182), and `activemq::transport::failover::FailoverTask` (p. 1936).

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/CompositeTask.h`

## 6.208 `activemq::threads::CompositeTaskRunner` Class Reference

A **Task** (p. 3846) Runner that can contain one or more `CompositeTasks` that are each checked for pending work and run if any is present in the order that the tasks were added.

```
#include <src/main/activemq/threads/CompositeTaskRunner.h>
```

Inheritance diagram for `activemq::threads::CompositeTaskRunner`:

### Public Member Functions

- **CompositeTaskRunner** ()
- virtual **~CompositeTaskRunner** ()
- void **addTask** (`CompositeTask` \*task)  
*Adds a new **CompositeTask** (p. 1255) to the Set of Tasks that this class manages.*
- void **removeTask** (`CompositeTask` \*task)  
*Removes a **CompositeTask** (p. 1255) that was added previously.*
- virtual void **shutdown** (unsigned int timeout)  
*Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.*
- virtual void **shutdown** ()  
*Shutdown once the task has finished and the TaskRunner's thread has exited.*
- virtual void **wakeup** ()  
*Signal the **TaskRunner** (p. 3849) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3846) instance will be run until its iterate method has returned false indicating it is done.*

### Protected Member Functions

- virtual void **run** ()  
*Run method - called by the Thread class in the context of the thread.*



- virtual bool **iterate** ()

*Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.*

### 6.208.1 Detailed Description

A **Task** (p. 3846) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.

Since

3.0

### 6.208.2 Constructor & Destructor Documentation

6.208.2.1 **activemq::threads::CompositeTaskRunner::CompositeTaskRunner** ( )

6.208.2.2 **virtual activemq::threads::CompositeTaskRunner::~~CompositeTaskRunner** ( )  
[virtual]

### 6.208.3 Member Function Documentation

6.208.3.1 **void activemq::threads::CompositeTaskRunner::addTask** ( **CompositeTask** \* *task* )

Adds a new **CompositeTask** (p. 1255) to the Set of Tasks that this class manages.

#### Parameters

<i>task</i>	- Pointer to a <b>CompositeTask</b> (p. 1255) instance.
-------------	---

6.208.3.2 **virtual bool activemq::threads::CompositeTaskRunner::iterate** ( )  
[protected, virtual]

Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

#### Returns

true if the task should be run again or false if the task has completed and the runner should wait for a wakeup call.

Implements **activemq::threads::Task** (p. 3847).

### 6.208.3.3 `void activemq::threads::CompositeTaskRunner::removeTask ( CompositeTask * task )`

Removes a **CompositeTask** (p. 1255) that was added previously.

#### Parameters

<i>task</i>	- Pointer to a <b>CompositeTask</b> (p. 1255) instance.
-------------	---

### 6.208.3.4 `virtual void activemq::threads::CompositeTaskRunner::run ( )` [protected, virtual]

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 3419).

### 6.208.3.5 `virtual void activemq::threads::CompositeTaskRunner::shutdown ( )` [virtual]

Shutdown once the task has finished and the TaskRunner's thread has exited.

Implements **activemq::threads::TaskRunner** (p. 3849).

### 6.208.3.6 `virtual void activemq::threads::CompositeTaskRunner::shutdown ( unsigned int timeout )` [virtual]

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

#### Parameters

<i>timeout</i>	- Time in Milliseconds to wait for the task to stop.
----------------	--

Implements **activemq::threads::TaskRunner** (p. 3849).

### 6.208.3.7 `virtual void activemq::threads::CompositeTaskRunner::wakeup ( )` [virtual]

Signal the **TaskRunner** (p. 3849) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3846) instance will be run until its iterate method has returned false indicating it is done.

Implements **activemq::threads::TaskRunner** (p. 3850).

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/CompositeTaskRunner.h`

## 6.209 activemq::transport::CompositeTransport Class Reference

A Composite **Transport** (p. 3996) is a **Transport** (p. 3996) implementation that is composed of several Transports.

```
#include <src/main/activemq/transport/CompositeTransport.h>
```

Inheritance diagram for activemq::transport::CompositeTransport:

### Public Member Functions

- virtual **~CompositeTransport** ()
- virtual void **addURI** (const **List**< **URI** > &uris)=0  
*Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3996) is a composite of.*
- virtual void **removeURI** (const **List**< **URI** > &uris)=0  
*Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3996) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3996) should result in that **Transport** (p. 3996) being disposed of.*

### 6.209.1 Detailed Description

A Composite **Transport** (p. 3996) is a **Transport** (p. 3996) implementation that is composed of several Transports. The composition could be such that only one **Transport** (p. 3996) exists for each URI that is composed or there could be many active Transports working at once.

Since

3.0

### 6.209.2 Constructor & Destructor Documentation

6.209.2.1 virtual activemq::transport::CompositeTransport::~~CompositeTransport ( )  
 [inline, virtual]

### 6.209.3 Member Function Documentation

6.209.3.1 virtual void activemq::transport::CompositeTransport::addURI ( const **List**< **URI** > & *uris* ) [pure virtual]

Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3996) is a composite of.

**Parameters**

<i>uris</i>	The new URI set to add to the set this composite maintains.
-------------	---

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1933).

**6.209.3.2** `virtual void activemq::transport::CompositeTransport::removeURI ( const List< URI > & uris ) [pure virtual]`

Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3996) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3996) should result in that **Transport** (p. 3996) being disposed of.

**Parameters**

<i>uris</i>	The new URI set to remove to the set this composite maintains.
-------------	--

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1938).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/CompositeTransport.h`

## 6.210 `decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >` > Class Template Reference

Interface for a **Map** (p. 2538) type that provides additional atomic `putIfAbsent`, `remove`, and `replace` methods alongside the already available **Map** (p. 2538) interface.

```
#include <src/main/decaf/util/concurrent/ConcurrentMap.h>
```

Inheritance diagram for `decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >`:

**Public Member Functions**

- `virtual ~ConcurrentMap ()`
- `virtual bool putIfAbsent (const K &key, const V &value)=0 throw ( decaf::lang::exceptions::Unsupported )`

*If the specified key is not already associated with a value, associate it with the given value.*

- `virtual bool remove (const K &key, const V &value)=0`

*Remove entry for key only if currently mapped to given value.*

- virtual bool **replace** (const K &key, const V &oldValue, const V &newValue)=0

*Replace entry for key only if currently mapped to given value.*

- virtual V **replace** (const K &key, const V &value)=0 throw ( decaf::lang::exceptions::NoSuchElementException )

*Replace entry for key only if currently mapped to some value.*

### 6.210.1 Detailed Description

template<typename K, typename V, typename COMPARATOR> class decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >

Interface for a **Map** (p. 2538) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 2538) interface.

Since

1.0

### 6.210.2 Constructor & Destructor Documentation

6.210.2.1 template<typename K, typename V, typename COMPARATOR> virtual  
decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR  
>::~~ConcurrentMap ( ) [inline, virtual]

### 6.210.3 Member Function Documentation

6.210.3.1 template<typename K, typename V, typename COMPARATOR>  
virtual bool decaf::util::concurrent::ConcurrentMap< K, V,  
COMPARATOR >::putIfAbsent ( const K & key, const V & value ) throw (   
decaf::lang::exceptions::UnsupportedOperationException ) [pure  
virtual]

If the specified key is not already associated with a value, associate it with the given value.

This is equivalent to

```
if( !map.containsKey( key ) ) {  
    map.put( key, value );  
    return true;  
} else {  
    return false;  
}
```

except that the action is performed atomically.

**Parameters**

<i>key</i>	The key to map the value to.
<i>value</i>	The value to map to the given key.

**Returns**

true if the put operation was performed otherwise return false which indicates there was a value previously mapped to the key.

**Exceptions**

<i>UnsupportedOperationException</i>	if the put operation is not supported by this map
--------------------------------------	---

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1276), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1276), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1276), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1276), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1276), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1276), and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1276).

**6.210.3.2** `template<typename K, typename V, typename COMPARATOR> virtual bool decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >::remove ( const K & key, const V & value ) [pure virtual]`

Remove entry for key only if currently mapped to given value.

Acts as

```
if( ( map.containsKey( key ) && ( map.get( key ) == value ) ) ) {
    map.remove( key );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

**Parameters**

<i>key</i>	key with which the specified value is associated.
<i>value</i>	value associated with the specified key.

## Returns

true if the value was removed, false otherwise

Implemented in **decaf::util::concurrent::ConcurrentStdMap< K, V, COMPARATOR >** (p. 1277), **decaf::util::concurrent::ConcurrentStdMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1277), **decaf::util::concurrent::ConcurrentStdMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1277), **decaf::util::concurrent::ConcurrentStdMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1277), **decaf::util::concurrent::ConcurrentStdMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1277), **decaf::util::concurrent::ConcurrentStdMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 1277), and **decaf::util::concurrent::ConcurrentStdMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1277).

6.210.3.3 **template<typename K, typename V, typename COMPARATOR> virtual bool**  
**decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >::replace (**  
**const K & key, const V & oldValue, const V & newValue )** [pure virtual]

Replace entry for key only if currently mapped to given value.

Acts as

```
if ( ( map.containsKey( key ) && ( map.get( key ) == oldValue ) ) ) {
    map.put( key, newValue );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

## Parameters

<i>key</i>	key with which the specified value is associated.
<i>oldValue</i>	value expected to be associated with the specified key.
<i>newValue</i>	value to be associated with the specified key.

## Returns

true if the value was replaced

Implemented in **decaf::util::concurrent::ConcurrentStdMap< K, V, COMPARATOR >** (p. 1279), **decaf::util::concurrent::ConcurrentStdMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1279), **decaf::util::concurrent::ConcurrentStdMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1279), **decaf::util::concurrent::ConcurrentStdMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1279), **decaf::util::concurrent::ConcurrentStdMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >**

(p. 1279), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1279), and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1279).

**6.210.3.4** `template<typename K, typename V, typename COMPARATOR>`  
`virtual V decaf::util::concurrent::ConcurrentMap< K, V,`  
`COMPARATOR >::replace ( const K & key, const V & value ) throw (`  
`decaf::lang::exceptions::NoSuchElementException ) [pure`  
`virtual]`

Replace entry for key only if currently mapped to some value.

Acts as

```
if( ( map.containsKey( key ) ) ) {
    return map.put( key, value );
} else {
    throw NoSuchElementException(...);
};
```

except that the action is performed atomically.

#### Parameters

<i>key</i>	key with which the specified value is associated.
<i>value</i>	value to be associated with the specified key.

#### Returns

copy of the previous value associated with specified key, or throws an `NoSuchElementException` if there was no mapping for key.

#### Exceptions

<i>NoSuchElementException</i>	if there was no previous mapping.
-------------------------------	-----------------------------------

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1278), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1278), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1278), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1278), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1278), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1278), and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1278).



The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/ConcurrentMap.h

## 6.211 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference

**Map** (p. 2538) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

```
#include <src/main/decaf/util/concurrent/ConcurrentStlMap.h>
```

Inheritance diagram for decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >:

### Public Member Functions

- **ConcurrentStlMap** ()  
*Default constructor - does nothing.*
- **ConcurrentStlMap** (const **ConcurrentStlMap** &source)  
*Copy constructor - copies the content of the given map into this one.*
- **ConcurrentStlMap** (const **Map**< K, V, COMPARATOR > &source)  
*Copy constructor - copies the content of the given map into this one.*
- virtual ~**ConcurrentStlMap** ()
- virtual bool **equals** (const **ConcurrentStlMap** &source) const
- virtual bool **equals** (const **Map**< K, V, COMPARATOR > &source) const  
*Comparison, equality is dependent on the method of determining if the element are equal.*
- virtual void **copy** (const **ConcurrentStlMap** &source)
- virtual void **copy** (const **Map**< K, V, COMPARATOR > &source)  
*Copies the content of the source map into this map.*
- virtual void **clear** () throw ( decaf::lang::exceptions::UnsupportedOperationException )  
*Removes all keys and values from this map.*  
*Exceptions*

UnsupportedOperation Exception	<i>if this map is unmodifiable.</i>
-----------------------------------	-------------------------------------

- virtual bool **containsKey** (const K &key) const

*Indicates whether or this map contains a value for the given key.*

**Parameters**

key	<i>The key to look up.</i>
-----	----------------------------

**Returns**

*true if this map contains the value, otherwise false.*

- virtual bool **containsValue** (const V &value) const

*Indicates whether or this map contains a value for the given value, i.e. they are equal, this is done by operator== so the types must pass equivalence testing in this manner.*

**Parameters**

value	<i>The Value to look up.</i>
-------	------------------------------

**Returns**

*true if this map contains the value, otherwise false.*

- virtual bool **isEmpty** () const

**Returns**

*if the **Map** (p. 2538) contains any element or not, TRUE or FALSE*

- virtual std::size\_t **size** () const

**Returns**

*The number of elements (key/value pairs) in this map.*

- virtual V & **get** (const K &key) throw ( lang::exceptions::NoSuchElementException )

*Gets the value mapped to the specified key in the **Map** (p. 2538).*

*If there is no element in the map whose key is equivalent to the key provided then a NoSuchElementException is thrown.*

**Parameters**

key	<i>The search key.</i>
-----	------------------------

**Returns**

*A reference to the value for the given key.*

**Exceptions**

NoSuchElementException	<i>if the key requests doesn't exist in the <b>Map</b> (p. 2538).</i>
------------------------	---

- virtual const V & **get** (const K &key) const throw ( lang::exceptions::NoSuchElementException )

)

*Gets the value mapped to the specified key in the **Map** (p. 2538).  
 If there is no element in the map whose key is equivalent to the key provided then a  
 NoSuchElementException is thrown.*

**Parameters**

key	The search key.
-----	-----------------

**Returns**

*A {const} reference to the value for the given key.*

**Exceptions**

NoSuchElementExcep- tion	if the key requests doesn't exist in the <b>Map</b> (p. 2538).
-----------------------------	--

- virtual void **put** (const K &key, const V &value) throw ( decaf::lang::exceptions::UnsupportedOperationException )

*Sets the value for the specified key.*

**Parameters**

key	The target key.
value	The value to be set.

**Exceptions**

UnsupportedOpera- tionException	if this map is unmodifiable.
------------------------------------	------------------------------

- virtual void **putAll** (const **ConcurrentStdMap**< K, V, COMPARATOR > &other) throw ( decaf::lang::exceptions::UnsupportedOperationException )

- virtual void **putAll** (const **Map**< K, V, COMPARATOR > &other) throw ( decaf::lang::exceptions::UnsupportedOperationException )

*Stores a copy of the Mappings contained in the other **Map** (p. 2538) in this one.*

**Parameters**

other	A <b>Map</b> (p. 2538) instance whose elements are to all be inserted in this <b>Map</b> (p. 2538).
-------	---

**Exceptions**

UnsupportedOpera- tionException	If the implementing class does not support the putAll operation.
------------------------------------	--

- virtual V **remove** (const K &key) throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )

*Removes the value (key/value pair) for the specified key from the map, returns a copy  
 of the value that was mapped to the key.*

**Parameters**

key	The search key.
-----	-----------------

**Returns**

*a copy of the element that was previously mapped to the given key*

**Exceptions**

NoSuchElementException	if this key is not in the <b>Map</b> (p. 2538).
UnsupportedOperationException	if this map is unmodifiable.

- virtual std::vector< K > **keySet** () const

*Returns a **Set** (p. 3538) view of the mappings contained in this map.*

*The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 2223), **Set.remove** (p. 168), removeAll, retainAll and clear operations. It does not support the add or addAll operations.*

**Returns**

*the entire set of keys in this map as a std::vector.*

- virtual std::vector< V > **values** () const

**Returns**

*the entire set of values in this map as a std::vector.*

- bool **putIfAbsent** (const K &key, const V &value) throw ( decaf::lang::exceptions::UnsupportedOperationException )

*If the specified key is not already associated with a value, associate it with the given value.*

- bool **remove** (const K &key, const V &value)

*Remove entry for key only if currently mapped to given value.*

- bool **replace** (const K &key, const V &oldValue, const V &newValue)

*Replace entry for key only if currently mapped to given value.*

- V **replace** (const K &key, const V &value) throw ( decaf::lang::exceptions::NoSuchElementException )

*Replace entry for key only if currently mapped to some value.*

- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )

*Locks the object.*

- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )

*Attempts to **Lock** (p. 2450) the object, if the lock is already held by another thread than this method returns false.*

- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )

*Unlocks the object.*

- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals a waiter on this object that it can now wake up and continue.*

- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals the waiters on this object that it can now wake up and continue.*

### 6.211.1 Detailed Description

```
template<typename K, typename V, typename COMPARATOR = std::less<K>> class decaf::util::concurrent::ConcurrentStlMap<
K, V, COMPARATOR >
```

**Map** (p. 2538) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map. This version of **Map** (p. 2538) extends the **ConcurrentMap** (p. 1260) interface and implements all the methods defined in that interface. Unlike a Java ConcurrentHashMap this implementation synchronizes all methods such that any call to this class will block if another thread is already holding a lock, much like the Java HashTable.

Since

1.0

### 6.211.2 Constructor & Destructor Documentation

```
6.211.2.1 template<typename K, typename V, typename COMPARATOR = std::less<K>>
decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
>::ConcurrentStlMap ( ) [inline]
```

Default constructor - does nothing.

```

6.211.2.2  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
            >::ConcurrentStlMap ( const ConcurrentStlMap< K, V, COMPARATOR > &
            source ) [inline]

```

Copy constructor - copies the content of the given map into this one.

#### Parameters

<i>source</i>	The source map.
---------------	-----------------

```

6.211.2.3  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
            >::ConcurrentStlMap ( const Map< K, V, COMPARATOR > & source )
            [inline]

```

Copy constructor - copies the content of the given map into this one.

#### Parameters

<i>source</i>	The source map.
---------------	-----------------

```

6.211.2.4  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
            >::~~ConcurrentStlMap ( ) [inline, virtual]

```

### 6.211.3 Member Function Documentation

```

6.211.3.1  template<typename K, typename V, typename
            COMPARATOR = std::less<K>> virtual void
            decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::clear (
            ) throw ( decaf::lang::exceptions::UnsupportedOperationException )
            [inline, virtual]

```

Removes all keys and values from this map.

#### Exceptions

<i>UnsupportedOperation Exception</i>	if this map is unmodifiable.
---	------------------------------

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2540).

Referenced by decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::copy().

6.211.3.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
 virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V,  
 COMPARATOR >::containsKey ( const K & key ) const [inline, virtual]`

Indicates whether or this map contains a value for the given key.

#### Parameters

<i>key</i>	The key to look up.
------------	---------------------

#### Returns

true if this map contains the value, otherwise false.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2541).

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putIfAbsent()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::remove()`, and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::replace()`.

6.211.3.3 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
 virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V,  
 COMPARATOR >::containsValue ( const V & value ) const [inline, virtual]`

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

#### Parameters

<i>value</i>	The Value to look up.
--------------	-----------------------

#### Returns

true if this map contains the value, otherwise false.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2542).

6.211.3.4 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
 virtual void decaf::util::concurrent::ConcurrentStlMap< K, V,  
 COMPARATOR >::copy ( const ConcurrentStlMap< K, V, COMPARATOR > &  
 source ) [inline, virtual]`

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::ConcurrentStlMap()`.

```
6.211.3.5  template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARATOR >::copy ( const Map< K, V, COMPARATOR > & source )
[inline, virtual]
```

Copies the content of the source map into this map.

Erases all existing data in this map.

#### Parameters

<i>source</i>	The source object to copy from.
---------------	---------------------------------

Implements **decaf::util::Map**< **K**, **V**, **COMPARATOR** > (p. 2543).

```
6.211.3.6  template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARATOR >::equals ( const Map< K, V, COMPARATOR > & source ) const
[inline, virtual]
```

Comparison, equality is dependent on the method of determining if the element are equal.

#### Parameters

<i>source</i>	- <b>Map</b> (p. 2538) to compare to this one.
---------------	--

#### Returns

true if the **Map** (p. 2538) passed is equal in value to this one.

Implements **decaf::util::Map**< **K**, **V**, **COMPARATOR** > (p. 2543).

```
6.211.3.7  template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARATOR >::equals ( const ConcurrentStlMap< K, V, COMPARATOR > &
source ) const [inline, virtual]
```

```
6.211.3.8  template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual V& decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
>::get ( const K & key ) throw ( lang::exceptions::NoSuchElementException
) [inline, virtual]
```

Gets the value mapped to the specified key in the **Map** (p. 2538).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** is thrown.

#### Parameters

<i>key</i>	The search key.
------------	-----------------



## Returns

A reference to the value for the given key.

## Exceptions

<i>NoSuchElementException</i>	if the key requests doesn't exist in the <b>Map</b> (p. 2538).
-------------------------------	--

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2543).

```
6.211.3.9  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual const V& decaf::util::concurrent::ConcurrentStlMap<
            K, V, COMPARATOR >::get ( const K & key ) const throw (
            lang::exceptions::NoSuchElementException ) [inline, virtual]
```

Gets the value mapped to the specified key in the **Map** (p. 2538).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** is thrown.

## Parameters

<i>key</i>	The search key.
------------	-----------------

## Returns

A {const} reference to the value for the given key.

## Exceptions

<i>NoSuchElementException</i>	if the key requests doesn't exist in the <b>Map</b> (p. 2538).
-------------------------------	--

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2544).

```
6.211.3.10 template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V,
            COMPARATOR >::isEmpty ( ) const [inline, virtual]
```

## Returns

if the **Map** (p. 2538) contains any element or not, TRUE or FALSE

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2545).

```
6.211.3.11 template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual std::vector<K> decaf::util::concurrent::ConcurrentStlMap< K, V,
            COMPARATOR >::keySet ( ) const [inline, virtual]
```

Returns a **Set** (p. 3538) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 2223), **Set.remove** (p. 168), removeAll, retainAll and clear operations. It does not support the add or addAll operations.

### Returns

the entire set of keys in this map as a std::vector.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2546).

```
6.211.3.12  template<typename K, typename V, typename
             COMPARATOR = std::less<K>> virtual void
             decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::lock (
             ) throw ( decaf::lang::exceptions::RuntimeException ) [inline,
             virtual]
```

Locks the object.

### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3812).

```
6.211.3.13  template<typename K, typename V, typename
             COMPARATOR = std::less<K>> virtual void
             decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
             >::notify ( ) throw ( decaf::lang::exceptions::RuntimeException,
             decaf::lang::exceptions::IllegalMonitorStateException ) [inline,
             virtual]
```

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3811) Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3813).

6.211.3.14 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
 virtual void decaf::util::concurrent::ConcurrentStlMap<  
 K, V, COMPARATOR >::notifyAll (    ) throw (  
 decaf::lang::exceptions::RuntimeException,  
 decaf::lang::exceptions::IllegalMonitorStateException ) [inline,  
 virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3811) Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3814).

6.211.3.15 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
 virtual void decaf::util::concurrent::ConcurrentStlMap< K,  
 V, COMPARATOR >::put ( const K & key, const V & value ) throw (  
 decaf::lang::exceptions::UnsupportedOperationException )  
 [inline, virtual]`

Sets the value for the specified key.

### Parameters

<i>key</i>	The target key.
<i>value</i>	The value to be set.

### Exceptions

<i>UnsupportedOperationException</i>	if this map is unmodifiable.
--------------------------------------	------------------------------

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2547).

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putAll()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putIfAbsent()`, and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::replace()`.

```

6.211.3.16  template<typename K, typename V, typename COMPARTOR = std::less<K>>
virtual void decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARTOR >::putAll ( const Map< K, V, COMPARTOR > & other )
throw ( decaf::lang::exceptions::UnsupportedOperationException )
[inline, virtual]

```

Stores a copy of the Mappings contained in the other **Map** (p. 2538) in this one.

#### Parameters

<i>other</i>	A <b>Map</b> (p. 2538) instance whose elements are to all be inserted in this <b>Map</b> (p. 2538).
--------------	---

#### Exceptions

<i>UnsupportedOperationException</i>	If the implementing class does not support the putAll operation.
--------------------------------------	--

Implements **decaf::util::Map< K, V, COMPARTOR >** (p. 2548).

```

6.211.3.17  template<typename K, typename V, typename COMPARTOR = std::less<K>>
virtual void decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARTOR >::putAll ( const ConcurrentStlMap< K, V, COMPARTOR > &
other ) throw ( decaf::lang::exceptions::UnsupportedOperationException
) [inline, virtual]

```

Referenced by **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARTOR >::copy()**.

```

6.211.3.18  template<typename K, typename V, typename COMPARTOR = std::less<K>>
bool decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARTOR >::putIfAbsent ( const K & key, const V & value ) throw
( decaf::lang::exceptions::UnsupportedOperationException )
[inline, virtual]

```

If the specified key is not already associated with a value, associate it with the given value.

This is equivalent to

```

if( !map.containsKey( key ) ) {
    map.put( key, value );
    return true;
} else {
    return false;
}

```

except that the action is performed atomically.

## 6.211 decaf::util::concurrent::ConcurrentStdMap< K, V, COMPARATOR > Class Template Reference 1277

### Parameters

<i>key</i>	The key to map the value to.
<i>value</i>	The value to map to the given key.

### Returns

true if the put operation was performed otherwise return false which indicates there was a value previously mapped to the key.

### Exceptions

<i>UnsupportedOperationException</i>	if the put operation is not supported by this map
--------------------------------------	---

Implements **decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >** (p. 1261).

**6.211.3.19** `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
bool decaf::util::concurrent::ConcurrentStdMap< K, V, COMPARATOR  
>::remove ( const K & key, const V & value ) [inline, virtual]`

Remove entry for key only if currently mapped to given value.

Acts as

```
if( map.containsKey( key ) && ( map.get( key ) == value ) ) {  
    map.remove( key );  
    return true;  
} else {  
    return false;  
}
```

except that the action is performed atomically.

### Parameters

<i>key</i>	key with which the specified value is associated.
<i>value</i>	value associated with the specified key.

### Returns

true if the value was removed, false otherwise

Implements **decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >** (p. 1262).

```
6.211.3.20  template<typename K, typename V, typename COMPARTOR =
std::less<K>> virtual V decaf::util::concurrent::ConcurrentStlMap<
K, V, COMPARTOR >::remove ( const K & key ) throw (
decaf::lang::exceptions::NoSuchElementException,
decaf::lang::exceptions::UnsupportedOperationException )
[inline, virtual]
```

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

#### Parameters

<i>key</i>	The search key.
------------	-----------------

#### Returns

a copy of the element that was previously mapped to the given key

#### Exceptions

<i>NoSuchElementException</i>	if this key is not in the <b>Map</b> (p. 2538).
<i>UnsupportedOperationException</i>	if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARTOR >** (p. 2548).

```
6.211.3.21  template<typename K, typename V, typename COMPARTOR =
std::less<K>> V decaf::util::concurrent::ConcurrentStlMap<
K, V, COMPARTOR >::replace ( const K & key, const V & value ) throw (
decaf::lang::exceptions::NoSuchElementException ) [inline,
virtual]
```

Replace entry for key only if currently mapped to some value.

Acts as

```
if( map.containsKey( key ) ) {
    return map.put( key, value );
} else {
    throw NoSuchElementException(...);
};
```

except that the action is performed atomically.

#### Parameters

<i>key</i>	key with which the specified value is associated.
<i>value</i>	value to be associated with the specified key.

## Returns

copy of the previous value associated with specified key, or throws an `NoSuchElementException` if there was no mapping for key.

## Exceptions

<i>NoSuchElementException</i>	if there was no previous mapping.
-------------------------------	-----------------------------------

Implements **decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >** (p. 1264).

**6.211.3.22** `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
bool decaf::util::concurrent::ConcurrentStdMap< K, V, COMPARATOR  
>::replace ( const K & key, const V & oldValue, const V & newValue )  
[inline, virtual]`

Replace entry for key only if currently mapped to given value.

Acts as

```
if( map.containsKey( key ) && ( map.get( key ) == oldValue ) ) {
    map.put( key, newValue );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

## Parameters

<i>key</i>	key with which the specified value is associated.
<i>oldValue</i>	value expected to be associated with the specified key.
<i>newValue</i>	value to be associated with the specified key.

## Returns

true if the value was replaced

Implements **decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >** (p. 1263).

**6.211.3.23** `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual std::size_t decaf::util::concurrent::ConcurrentStdMap< K, V,  
COMPARATOR >::size ( ) const [inline, virtual]`

## Returns

The number of elements (key/value pairs) in this map.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2549).

```
6.211.3.24  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::concurrent::ConcurrentStlMap<
            K, V, COMPARATOR >::tryLock (    ) throw (
            decaf::lang::exceptions::RuntimeException ) [inline, virtual]
```

Attempts to **Lock** (p. 2450) the object, if the lock is already held by another thread than this method returns false.

### Returns

true if the lock was acquired, false if it is already held by another thread.

### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3815).

```
6.211.3.25  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual void decaf::util::concurrent::ConcurrentStlMap<
            K, V, COMPARATOR >::unlock (    ) throw (
            decaf::lang::exceptions::RuntimeException ) [inline, virtual]
```

Unlocks the object.

### Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3816).

```
6.211.3.26  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual std::vector<V> decaf::util::concurrent::ConcurrentStlMap< K, V,
            COMPARATOR >::values (    ) const [inline, virtual]
```

### Returns

the entire set of values in this map as a std::vector.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2550).

Referenced by decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::values().



6.211.3.27 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
 virtual void decaf::util::concurrent::ConcurrentStlMap<  
 K, V, COMPARATOR >::wait ( long long millisecs ) throw  
 ( decaf::lang::exceptions::RuntimeException,  
 decaf::lang::exceptions::IllegalMonitorStateException,  
 decaf::lang::exceptions::InterruptedException ) [inline,  
 virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3811) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3819).

6.211.3.28 `template<typename K, typename V, typename  
 COMPARATOR = std::less<K>> virtual void  
 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR  
 >::wait ( ) throw ( decaf::lang::exceptions::RuntimeException,  
 decaf::lang::exceptions::IllegalMonitorStateException,  
 decaf::lang::exceptions::InterruptedException ) [inline,  
 virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3811) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3818).

```

6.211.3.29  template<typename K, typename V, typename COMPARATOR = std::less<K>>
              virtual void decaf::util::concurrent::ConcurrentStlMap<
                K, V, COMPARATOR >::wait ( long long millisecs, int nanos
                ) throw ( decaf::lang::exceptions::RuntimeException,
                decaf::lang::exceptions::IllegalArgumentException,
                decaf::lang::exceptions::IllegalMonitorStateException,
                decaf::lang::exceptions::InterruptedException ) [inline,
                virtual]

```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INIFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

#### Exceptions

<i>IllegalArgumentEx- ception</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorState- Exception</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3811) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3820).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/ConcurrentStlMap.h

## 6.212 decaf::util::concurrent::locks::Condition Class Reference

**Condition** (p. 1282) factors out the **Mutex** (p. 2866) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 2452) implementations.

```
#include <src/main/decaf/util/concurrent/locks/Condition.h>
```

## Public Member Functions

- virtual `~Condition()`
- virtual void **await** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Causes the current thread to wait until it is signaled or interrupted.*
- virtual void **awaitUninterruptibly** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Causes the current thread to wait until it is signalled.*
- virtual long long **awaitNanos** (long long nanosTimeout)=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.*
- virtual bool **await** (long long time, const **TimeUnit** &unit)=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.*
- virtual bool **awaitUntil** (const **Date** &deadline)=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException )
- virtual void **signal** ()=0 throw ( decaf::lang::exceptions::RuntimeException )  
*Wakes up one waiting thread.*
- virtual void **signalAll** ()=0 throw ( decaf::lang::exceptions::RuntimeException )  
*Wakes up all waiting threads.*

### 6.212.1 Detailed Description

**Condition** (p. 1282) factors out the **Mutex** (p. 2866) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 2452) implementations. Where a **Lock** (p. 2452) replaces the use of synchronized statements, a **Condition** (p. 1282) replaces the use of the Object monitor methods.

Conditions (also known as condition queues or condition variables) provide a means for one thread to suspend execution (to "wait") until notified by another thread that some state condition may now be true. Because access to this shared state information occurs in different threads, it must be protected, so a lock of some form is associated

with the condition. The key property that waiting for a condition provides is that it atomically releases the associated lock and suspends the current thread.

A **Condition** (p. 1282) instance is intrinsically bound to a lock. To obtain a **Condition** (p. 1282) instance for a particular **Lock** (p. 2452) instance use its `newCondition()` method.

As an example, suppose we have a bounded buffer which supports `put` and `take` methods. If a `take` is attempted on an empty buffer, then the thread will block until an item becomes available; if a `put` is attempted on a full buffer, then the thread will block until a space becomes available. We would like to keep waiting `put` threads and `take` threads in separate wait-sets so that we can use the optimization of only notifying a single thread at a time when items or spaces become available in the buffer. This can be achieved using two **Condition** (p. 1282) instances.

```
class BoundedBuffer { Lock* lock = new ReentrantLock(); Condition* notFull = lock->newCondition(); Condition* notEmpty = lock->newCondition();
```

```
Object* items = new Object[100]; int putptr, takeptr, count;
```

```
public void put( Object* x ) throw( InterruptedException ) { lock->lock(); try { while( count == 100 ) notFull->await() (p. 1285); items[putptr] = x; if (++putptr == 100) putptr = 0; ++count; notEmpty->signal() (p. 1289); } catch(...) { lock->unlock(); } }
```

```
public Object take() throw( InterruptedException ) { lock->lock(); try { while(count == 0) notEmpty->await() (p. 1285); Object x = items[takeptr]; if (++takeptr == 100) takeptr = 0; --count; notFull->signal() (p. 1289); return x; } catch(...) { lock->unlock(); } } }
```

(The `ArrayBlockingQueue` class provides this functionality, so there is no reason to implement this sample usage class.)

#### Implementation Considerations

When waiting upon a **Condition** (p. 1282), a "spurious wakeup" is permitted to occur, in general, as a concession to the underlying platform semantics. This has little practical impact on most application programs as a **Condition** (p. 1282) should always be waited upon in a loop, testing the state predicate that is being waited for. An implementation is free to remove the possibility of spurious wakeups but it is recommended that applications programmers always assume that they can occur and so always wait in a loop.

The three forms of condition waiting (interruptible, non-interruptible, and timed) may differ in their ease of implementation on some platforms and in their performance characteristics. In particular, it may be difficult to provide these features and maintain specific semantics such as ordering guarantees. Further, the ability to interrupt the actual suspension of the thread may not always be feasible to implement on all platforms.

Consequently, an implementation is not required to define exactly the same guarantees or semantics for all three forms of waiting, nor is it required to support interruption of the actual suspension of the thread.

An implementation is required to clearly document the semantics and guarantees provided by each of the waiting methods, and when an implementation does support interruption of thread suspension then it must obey the interruption semantics as defined in this interface.

As interruption generally implies cancellation, and checks for interruption are often infrequent, an implementation can favor responding to an interrupt over normal method return. This is true even if it can be shown that the interrupt occurred after another action may have unblocked the thread. An implementation should document this behavior.

### Since

1.0

## 6.212.2 Constructor & Destructor Documentation

**6.212.2.1** `virtual decaf::util::concurrent::locks::Condition::~~Condition ( ) [inline, virtual]`

## 6.212.3 Member Function Documentation

**6.212.3.1** `virtual void decaf::util::concurrent::locks::Condition::await ( )  
throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::InterruptedException,  
decaf::lang::exceptions::IllegalMonitorStateException ) [pure  
virtual]`

Causes the current thread to wait until it is signaled or interrupted.

The lock associated with this **Condition** (p. 1282) is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

- \* Some other thread invokes the **signal()** (p. 1289) method for this **Condition** (p. 1282) and the current thread happens to be chosen as the thread to be awakened; or
- \* Some other thread invokes the **signalAll()** (p. 1289) method for this **Condition** (p. 1282); or
- \* Some other thread interrupts the current thread, and interruption of thread suspension is supported; or
- \* A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread:

- \* has its interrupted status set on entry to this method; or
- \* is interrupted while waiting and interruption of thread suspension is supported,

then **InterruptedException** is thrown and the current thread's interrupted status is cleared. It is not specified, in the first case, whether or not the test for interruption occurs before the lock is released.

### Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p. 1282) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as **IllegalMonitorStateException**) and the implementation must document that fact.

An implementation can favor responding to an interrupt over normal method return in response to a signal. In that case the implementation must ensure that the signal is redirected to another waiting thread, if there is one.

### Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while trying to wait on the <b>Condition</b> (p. 1282).
<i>InterruptedException</i>	if the current thread is interrupted (and interruption of thread suspension is supported)
<i>IllegalMonitorStateException</i>	if the caller is not the lock owner.

6.212.3.2 `virtual bool decaf::util::concurrent::locks::Condition::await ( long long time, const TimeUnit & unit ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException ) [pure virtual]`

Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.

This method is behaviorally equivalent to:

`awaitNanos(unit.toNanos(time)) > 0`

### Parameters

<i>time</i>	- the maximum time to wait
<i>unit</i>	- the time unit of the time argument

### Returns

false if the waiting time detectably elapsed before return from the method, else true

### Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while trying to wait on the <b>Condition</b> (p. 1282).
<i>InterruptedException</i>	if the current thread is interrupted (and interruption of thread suspension is supported)
<i>IllegalMonitorStateException</i>	if the caller is not the lock owner.

```
6.212.3.3 virtual long long decaf::util::concurrent::locks::Condition::awaitNanos ( long long
nanosTimeout ) throw ( decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::InterruptedException,
decaf::lang::exceptions::IllegalMonitorStateException ) [pure
virtual]
```

Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.

The lock associated with this condition is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of five things happens:

- \* Some other thread invokes the **signal()** (p. 1289) method for this **Condition** (p. 1282) and the current thread happens to be chosen as the thread to be awakened; or
- \* Some other thread invokes the **signalAll()** (p. 1289) method for this **Condition** (p. 1282); or
- \* Some other thread interrupts the current thread, and interruption of thread suspension is supported; or
- \* The specified waiting time elapses; or
- \* A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread:

- \* has its interrupted status set on entry to this method; or
- \* is interrupted while waiting and interruption of thread suspension is supported,

then **InterruptedException** is thrown and the current thread's interrupted status is cleared. It is not specified, in the first case, whether or not the test for interruption occurs before the lock is released.

The method returns an estimate of the number of nanoseconds remaining to wait given the supplied **nanosTimeout** value upon return, or a value less than or equal to zero if it timed out. This value can be used to determine whether and how long to re-wait in cases where the wait returns but an awaited condition still does not hold. Typical uses of this method take the following form:

```
synchronized boolean aMethod( long timeout, const TimeUnit& unit ) { long nanos-
Timeout = unit.toNanos(timeout); while (!conditionBeingWaitedFor) { if (nanosTime-
out > 0) nanosTimeout = theCondition->awaitNanos(nanosTimeout); else return false;
} // ... }
```

Design note: This method requires a nanosecond argument so as to avoid truncation errors in reporting remaining times. Such precision loss would make it difficult for programmers to ensure that total waiting times are not systematically shorter than specified when re-waits occur.

#### Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p. 1282) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as **IllegalMonitorStateException**) and the implementation must document that fact.

An implementation can favor responding to an interrupt over normal method return

in response to a signal, or over indicating the elapse of the specified waiting time. In either case the implementation must ensure that the signal is redirected to another waiting thread, if there is one.

### Parameters

<i>nanosTimeout</i>	- the maximum time to wait, in nanoseconds
---------------------	--

### Returns

an estimate of the nanosTimeout value minus the time spent waiting upon return from this method. A positive value may be used as the argument to a subsequent call to this method to finish waiting out the desired time. A value less than or equal to zero indicates that no time remains.

### Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while trying to wait on the <b>Condition</b> (p. 1282).
<i>InterruptedException</i>	if the current thread is interrupted (and interruption of thread suspension is supported)
<i>IllegalMonitorStateException</i>	if the caller is not the lock owner.

**6.212.3.4** `virtual void decaf::util::concurrent::locks::Condition::awaitUninterruptibly ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException ) [pure virtual]`

Causes the current thread to wait until it is signalled.

The lock associated with this condition is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

- \* Some other thread invokes the **signal()** (p. 1289) method for this **Condition** (p. 1282) and the current thread happens to be chosen as the thread to be awakened; or
- \* Some other thread invokes the **signalAll()** (p. 1289) method for this **Condition** (p. 1282); or
- \* A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread's interrupted status is set when it enters this method, or it is interrupted while waiting, it will continue to wait until signalled. When it finally returns from this method its interrupted status will still be set.

### Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p. 1282) when this method is called. It is up to the implementation to determine if this is the



case and if not, how to respond. Typically, an exception will be thrown (such as `IllegalMonitorStateException`) and the implementation must document that fact.

### Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while trying to wait on the <b>Condition</b> (p. 1282).
<i>IllegalMonitorStateException</i>	if the caller is not the lock owner.

**6.212.3.5** `virtual bool decaf::util::concurrent::locks::Condition::awaitUntil ( const Date & deadline ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException ) [pure virtual]`

**6.212.3.6** `virtual void decaf::util::concurrent::locks::Condition::signal ( ) throw ( decaf::lang::exceptions::RuntimeException ) [pure virtual]`

Wakes up one waiting thread.

If any threads are waiting on this condition then one is selected for waking up. That thread must then re-acquire the lock before returning from await.

### Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while trying to wait on the <b>Condition</b> (p. 1282).
-------------------------	---

**6.212.3.7** `virtual void decaf::util::concurrent::locks::Condition::signalAll ( ) throw ( decaf::lang::exceptions::RuntimeException ) [pure virtual]`

Wakes up all waiting threads.

If any threads are waiting on this condition then they are all woken up. Each thread must re-acquire the lock before it can return from await.

### Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while trying to wait on the <b>Condition</b> (p. 1282).
-------------------------	---

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/Condition.h`

## 6.213 decaf::util::concurrent::ConditionHandle Class Reference

```
#include <src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h>
```

### Public Member Functions

- **ConditionHandle ()**
- **~ConditionHandle ()**
- **ConditionHandle ()**
- **~ConditionHandle ()**

### Data Fields

- pthread\_cond\_t **condition**
- **MutexHandle \* mutex**
- HANDLE **semaphore**
- CRITICAL\_SECTION **criticalSection**
- volatile unsigned int **numWaiting**
- volatile unsigned int **numWake**
- volatile unsigned int **generation**

### 6.213.1 Constructor & Destructor Documentation

- 6.213.1.1 `decaf::util::concurrent::ConditionHandle::ConditionHandle ( )` [inline]
- 6.213.1.2 `decaf::util::concurrent::ConditionHandle::~~ConditionHandle ( )` [inline]
- 6.213.1.3 `decaf::util::concurrent::ConditionHandle::ConditionHandle ( )` [inline]
- 6.213.1.4 `decaf::util::concurrent::ConditionHandle::~~ConditionHandle ( )` [inline]

### 6.213.2 Field Documentation

- 6.213.2.1 `pthread_cond_t decaf::util::concurrent::ConditionHandle::condition`
- 6.213.2.2 `CRITICAL_SECTION decaf::util::concurrent::ConditionHandle::criticalSection`
- 6.213.2.3 `volatile unsigned int decaf::util::concurrent::ConditionHandle::generation`
- 6.213.2.4 `MutexHandle * decaf::util::concurrent::ConditionHandle::mutex`
- 6.213.2.5 `volatile unsigned int decaf::util::concurrent::ConditionHandle::numWaiting`
- 6.213.2.6 `volatile unsigned int decaf::util::concurrent::ConditionHandle::numWake`
- 6.213.2.7 `HANDLE decaf::util::concurrent::ConditionHandle::semaphore`

The documentation for this class was generated from the following files:

- `src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h`
- `src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h`

## 6.214 decaf::internal::util::concurrent::ConditionImpl Class Reference

```
#include <src/main/decaf/internal/util/concurrent/ConditionImpl.h>
```

### Static Public Member Functions

- static `decaf::util::concurrent::ConditionHandle * create (decaf::util::concurrent::MutexHandle *mutex)`  
*Creates the Condition object and attaches it to the given MutexHandle.*
- static void `destroy (decaf::util::concurrent::ConditionHandle *handle)`  
*Destroy a previously create Condition instance.*
- static void `wait (decaf::util::concurrent::ConditionHandle *condition)`

*Waits for the condition to be signaled.*

- static void **wait** (**decaf::util::concurrent::ConditionHandle** \*condition, long long mills, long long nanos)

*Waits for the condition to be signaled or for the time specified to ellapse.*

- static void **notify** (**decaf::util::concurrent::ConditionHandle** \*condition)

*Signals one Thread that is waiting on this condition to wake up.*

- static void **notifyAll** (**decaf::util::concurrent::ConditionHandle** \*condition)

*Signals all Threads that is waiting on this condition to wake up.*

### 6.214.1 Member Function Documentation

**6.214.1.1 static decaf::util::concurrent::ConditionHandle\* decaf::internal::util::concurrent::ConditionImpl::create ( decaf::util::concurrent::MutexHandle \* mutex ) [static]**

Creates the Condition object and attaches it to the given MutexHandle.

#### Parameters

<i>mutex</i>	the Mutex handle that this Condition is attached to.
--------------	--

#### Returns

a newly constructed Condition handle that is attached to the given handle.

**6.214.1.2 static void decaf::internal::util::concurrent::ConditionImpl::destroy ( decaf::util::concurrent::ConditionHandle \* handle ) [static]**

Destroy a previously create Condition instance.

#### Parameters

<i>handle</i>	The Condition handle to be destroyed.
---------------	---------------------------------------

**6.214.1.3 static void decaf::internal::util::concurrent::ConditionImpl::notify ( decaf::util::concurrent::ConditionHandle \* condition ) [static]**

Signals one Thread that is waiting on this condition to wake up.

#### Parameters

<i>condition</i>	the handle to the condition to wait on.
------------------	---

**6.214.1.4** `static void decaf::internal::util::concurrent::ConditionImpl::notifyAll ( decaf::util::concurrent::ConditionHandle * condition )` `[static]`

Signals all Threads that is waiting on this condition to wake up.

#### Parameters

<i>condition</i>	the handle to the condition to wait on.
------------------	---

**6.214.1.5** `static void decaf::internal::util::concurrent::ConditionImpl::wait ( decaf::util::concurrent::ConditionHandle * condition, long long mills, long long nanos )` `[static]`

Waits for the condition to be signaled or for the time specified to ellapse.

#### Parameters

<i>condition</i>	the handle to the condition to wait on.
<i>mills</i>	the time in milliseconds to wait for the condition to be signaled.
<i>nanos</i>	additional time in nanoseconds to wait for the thread to be signaled.

**6.214.1.6** `static void decaf::internal::util::concurrent::ConditionImpl::wait ( decaf::util::concurrent::ConditionHandle * condition )` `[static]`

Waits for the condition to be signaled.

#### Parameters

<i>condition</i>	the handle to the condition to wait on.
------------------	---

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/concurrent/ConditionImpl.h`

## 6.215 decaf::net::ConnectException Class Reference

```
#include <src/main/decaf/net/ConnectException.h>
```

Inheritance diagram for decaf::net::ConnectException:

### Public Member Functions

- `ConnectException ()` throw `()`

*Default Constructor.*

- **ConnectException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **ConnectException** (const **ConnectException** &ex) throw ()  
*Copy Constructor.*
- **ConnectException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **ConnectException** (const std::exception \*cause) throw ()  
*Constructor.*
- **ConnectException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **ConnectException** \* clone () const  
*Clones this exception.*
- virtual ~**ConnectException** () throw ()

## 6.215.1 Constructor & Destructor Documentation

6.215.1.1 **decaf::net::ConnectException::ConnectException ( ) throw ()** [inline]

Default Constructor.

6.215.1.2 **decaf::net::ConnectException::ConnectException ( const Exception & ex ) throw ()** [inline]

Conversion Constructor from some other Exception.

### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.215.1.3 **decaf::net::ConnectException::ConnectException ( const ConnectException & ex ) throw ()** [inline]

Copy Constructor.

**Parameters**

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.215.1.4** `decaf::net::ConnectException::ConnectException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.215.1.5** `decaf::net::ConnectException::ConnectException ( const std::exception * cause )` `throw ()` `[inline]`

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.215.1.6** `decaf::net::ConnectException::ConnectException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.215.1.7 **virtual** `decaf::net::ConnectException::~~ConnectException ( ) throw ( )`  
`[inline, virtual]`

## 6.215.2 Member Function Documentation

6.215.2.1 **virtual** `ConnectException*` `decaf::net::ConnectException::clone ( ) const`  
`[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

a new `Exception` instance that is a copy of this `Exception` object.

Reimplemented from `decaf::net::SocketException` (p. 3629).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ConnectException.h`

## 6.216 cms::Connection Class Reference

The client's connection to its provider.

```
#include <src/main/cms/Connection.h>
```

Inheritance diagram for `cms::Connection`:

### Public Member Functions

- **virtual** `~Connection ( )`
- **virtual** `void close ()=0 throw ( CMSException )`  
*Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).*
- **virtual** `const ConnectionMetaData * getMetaData () const =0 throw ( CMSException )`  
*Gets the metadata for this connection.*
- **virtual** `Session * createSession ()=0 throw ( CMSException )`  
*Creates an AUTO\_ACKNOWLEDGE Session (p. 3460).*
- **virtual** `Session * createSession (Session::AcknowledgeMode ackMode)=0 throw ( CMSException )`



*Creates a new **Session** (p. 3460) to work for this **Connection** (p. 1296) using the specified acknowledgment mode.*

- virtual std::string **getClientID** () const =0

*Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the **setClientID** method.*

- virtual void **setClientID** (const std::string &clientID)=0

*Sets the client identifier for this connection.*

- virtual **ExceptionListener** \* **getExceptionListener** () const =0

*Gets the registered Exception Listener for this connection.*

- virtual void **setExceptionListener** (**ExceptionListener** \*listener)=0

*Sets the registered Exception Listener for this connection.*

### 6.216.1 Detailed Description

The client's connection to its provider. Connections support concurrent use.

A connection serves several purposes:

- It encapsulates an open connection with a JMS provider. It typically represents an open TCP/IP socket between a client and the service provider software.
- Its creation is where client authentication takes place.
- It can specify a unique client identifier.
- It provides a **ConnectionMetaData** (p. 1425) object.
- It supports an optional **ExceptionListener** (p. 1893) object.

Because the creation of a connection involves setting up authentication and communication, a connection is a relatively heavy-weight object. Most clients will do all their messaging with a single connection. Other more advanced applications may use several connections. The CMS API does not architect a reason for using multiple connections; however, there may be operational reasons for doing so.

A CMS client typically creates a connection, one or more sessions, and a number of message producers and consumers. When a connection is created, it is in stopped mode. That means that no messages are being delivered.

It is typical to leave the connection in stopped mode until setup is complete (that is, until all message consumers have been created). At that point, the client calls the connection's start method, and messages begin arriving at the connection's consumers. This setup convention minimizes any client confusion that may result from asynchronous message delivery while the client is still in the process of setting itself up.

A connection can be started immediately, and the setup can be done afterwards. Clients that do this must be prepared to handle asynchronous message delivery while they are still in the process of setting up.

A message producer can send messages while a connection is stopped.

#### Since

1.0

### 6.216.2 Constructor & Destructor Documentation

**6.216.2.1** `virtual cms::Connection::~~Connection ( ) [inline, virtual]`

### 6.216.3 Member Function Documentation

**6.216.3.1** `virtual void cms::Connection::close ( ) throw ( CMSEException ) [pure virtual]`

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

#### Exceptions

<i><b>CMSEException</b></i> (p. 1190)
--

Implements **cms::Closeable** (p. 1180).

Implemented in **activemq::core::ActiveMQConnection** (p. 267).

**6.216.3.2** `virtual Session* cms::Connection::createSession ( Session::AcknowledgeMode ackMode ) throw ( CMSEException ) [pure virtual]`

Creates a new **Session** (p. 3460) to work for this **Connection** (p. 1296) using the specified acknowledgment mode.

#### Parameters

<i>ackMode</i>	the Acknowledgment Mode to use.
----------------	---------------------------------

#### Exceptions

<i><b>CMSEException</b></i> (p. 1190)
--

Implemented in **activemq::core::ActiveMQConnection** (p. 267).

**6.216.3.3** `virtual Session* cms::Connection::createSession ( ) throw ( CMSEException )`  
`[pure virtual]`

Creates an AUTO\_ACKNOWLEDGE Session (p. 3460).

#### Exceptions

<i><b>CMSEException</b></i> (p. 1190)	
--	--

Implemented in **activemq::core::ActiveMQConnection** (p. 268).

**6.216.3.4** `virtual std::string cms::Connection::getClientID ( ) const` `[pure virtual]`

Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the setClientID method.

#### Returns

Client Id String for this **Connection** (p. 1296).

#### Exceptions

<i><b>CMSEException</b></i> (p. 1190)	if the provider fails to return the client id or an internal error occurs.
--	--

Implemented in **activemq::core::ActiveMQConnection** (p. 270).

**6.216.3.5** `virtual ExceptionListener* cms::Connection::getExceptionListener ( ) const`  
`[pure virtual]`

Gets the registered Exception Listener for this connection.

#### Returns

pointer to an exception listener or NULL

Implemented in **activemq::core::ActiveMQConnection** (p. 270).

**6.216.3.6** `virtual const ConnectionMetaData* cms::Connection::getMetaData ( ) const`  
`throw ( CMSEException )` `[pure virtual]`

Gets the metadata for this connection.

#### Returns

the connection MetaData pointer ( caller does not own it ).

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	if the provider fails to get the connection metadata for this connection.
---	---

**See also**

**ConnectionMetaData** (p. 1425)

**Since**

2.0

Implemented in **activemq::core::ActiveMQConnection** (p. 271).

**6.216.3.7** `virtual void cms::Connection::setClientID ( const std::string & clientID )` [pure virtual]

Sets the client identifier for this connection.

The preferred way to assign a CMS client's client identifier is for it to be configured in a client-specific **ConnectionFactory** (p. 1361) object and transparently assigned to the **Connection** (p. 1296) object it creates.

If a client sets the client identifier explicitly, it must do so immediately after it creates the connection and before any other action on the connection is taken. After this point, setting the client identifier is a programming error that should throw an **IllegalStateException** (p. 2059).

**Parameters**

<i>clientID</i>	The unique client identifier to assign to the <b>Connection</b> (p. 1296).
-----------------	--

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	if the provider fails to set the client id due to some internal error.
<b><i>InvalidClientIDException</i></b>	if the id given is somehow invalid or is a duplicate.
<b><i>IllegalStateException</i></b> (p. 2059)	if the client tries to set the id after a <b>Connection</b> (p. 1296) method has been called.

Implemented in **activemq::core::ActiveMQConnection** (p. 277).

**6.216.3.8** `virtual void cms::Connection::setExceptionListener ( ExceptionListener * listener )` [pure virtual]

Sets the registered Exception Listener for this connection.

**Parameters**

<i>listener</i>	pointer to and <b>ExceptionListener</b> (p. 1893)
-----------------	---

Implemented in **activemq::core::ActiveMQConnection** (p. 278).

The documentation for this class was generated from the following file:

- src/main/cms/Connection.h

**6.217 activemq::commands::ConnectionControl Class Reference**

```
#include <src/main/activemq/commands/ConnectionControl.h>
```

Inheritance diagram for activemq::commands::ConnectionControl:

**Public Member Functions**

- **ConnectionControl** ()
- virtual **~ConnectionControl** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ConnectionControl \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual bool **isClose** () const
- virtual void **setClose** (bool close)
- virtual bool **isExit** () const
- virtual void **setExit** (bool exit)
- virtual bool **isFaultTolerant** () const
- virtual void **setFaultTolerant** (bool faultTolerant)

- virtual bool **isResume** () const
- virtual void **setResume** (bool **resume**)
- virtual bool **isSuspend** () const
- virtual void **setSuspend** (bool **suspend**)
- virtual const std::string & **getConnectedBrokers** () const
- virtual std::string & **getConnectedBrokers** ()
- virtual void **setConnectedBrokers** (const std::string &**connectedBrokers**)
- virtual const std::string & **getReconnectTo** () const
- virtual std::string & **getReconnectTo** ()
- virtual void **setReconnectTo** (const std::string &**reconnectTo**)
- virtual bool **isRebalanceConnection** () const
- virtual void **setRebalanceConnection** (bool **rebalanceConnection**)
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_CONNECTIONCONTROL** = 18

### Protected Attributes

- bool **close**
- bool **exit**
- bool **faultTolerant**
- bool **resume**
- bool **suspend**
- std::string **connectedBrokers**
- std::string **reconnectTo**
- bool **rebalanceConnection**

### 6.217.1 Constructor & Destructor Documentation

6.217.1.1 `activemq::commands::ConnectionControl::ConnectionControl ( )`

6.217.1.2 `virtual activemq::commands::ConnectionControl::~~ConnectionControl ( )`  
[virtual]

### 6.217.2 Member Function Documentation

6.217.2.1 `virtual ConnectionControl* activemq::commands::ConnectionControl::cloneDataStructure ( )`  
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1714).

6.217.2.2 `virtual void activemq::commands::ConnectionControl::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<code>src</code>	- Source Object
------------------	-----------------

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

6.217.2.3 `virtual bool activemq::commands::ConnectionControl::equals ( const DataStructure * value ) const` [virtual]

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

6.217.2.4 `virtual const std::string& activemq::commands::ConnectionControl::getConnectedBrokers ( ) const [virtual]`

6.217.2.5 `virtual std::string& activemq::commands::ConnectionControl::getConnectedBrokers ( ) [virtual]`

6.217.2.6 `virtual unsigned char activemq::commands::ConnectionControl::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

### Returns

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).



- 6.217.2.7 `virtual const std::string& activemq::commands::ConnectionControl::getReconnectTo ( ) const [virtual]`
- 6.217.2.8 `virtual std::string& activemq::commands::ConnectionControl::getReconnectTo ( ) [virtual]`
- 6.217.2.9 `virtual bool activemq::commands::ConnectionControl::isClose ( ) const [virtual]`
- 6.217.2.10 `virtual bool activemq::commands::ConnectionControl::isExit ( ) const [virtual]`
- 6.217.2.11 `virtual bool activemq::commands::ConnectionControl::isFaultTolerant ( ) const [virtual]`
- 6.217.2.12 `virtual bool activemq::commands::ConnectionControl::isRebalanceConnection ( ) const [virtual]`
- 6.217.2.13 `virtual bool activemq::commands::ConnectionControl::isResume ( ) const [virtual]`
- 6.217.2.14 `virtual bool activemq::commands::ConnectionControl::isSuspend ( ) const [virtual]`
- 6.217.2.15 `virtual void activemq::commands::ConnectionControl::setClose ( bool close ) [virtual]`
- 6.217.2.16 `virtual void activemq::commands::ConnectionControl::setConnectedBrokers ( const std::string & connectedBrokers ) [virtual]`
- 6.217.2.17 `virtual void activemq::commands::ConnectionControl::setExit ( bool exit ) [virtual]`
- 6.217.2.18 `virtual void activemq::commands::ConnectionControl::setFaultTolerant ( bool faultTolerant ) [virtual]`
- 6.217.2.19 `virtual void activemq::commands::ConnectionControl::setRebalanceConnection ( bool rebalanceConnection ) [virtual]`
- 6.217.2.20 `virtual void activemq::commands::ConnectionControl::setReconnectTo ( const std::string & reconnectTo ) [virtual]`
- 6.217.2.21 `virtual void activemq::commands::ConnectionControl::setResume ( bool resume ) [virtual]`
- 6.217.2.22 `virtual void activemq::commands::ConnectionControl::setSuspend ( bool suspend ) [virtual]`
- 6.217.2.23 `virtual std::string activemq::commands::ConnectionControl::toString ( ) const [virtual]`

**Returns**

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 770).

**6.217.2.24** `virtual Pointer<Command> activemq::commands::ConnectionControl::visit  
( activemq::state::CommandVisitor * visitor ) throw (  
exceptions::ActiveMQException )` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

**Returns**

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 1232).

**6.217.3 Field Documentation**

**6.217.3.1** `bool activemq::commands::ConnectionControl::close` [protected]

**6.217.3.2** `std::string activemq::commands::ConnectionControl::connectedBrokers`  
[protected]

**6.217.3.3** `bool activemq::commands::ConnectionControl::exit` [protected]

**6.217.3.4** `bool activemq::commands::ConnectionControl::faultTolerant`  
[protected]

**6.217.3.5** `const unsigned char activemq::commands::ConnectionControl::ID_-  
CONNECTIONCONTROL = 18` [static]

**6.217.3.6** `bool activemq::commands::ConnectionControl::rebalanceConnection`  
[protected]

**6.217.3.7** `std::string activemq::commands::ConnectionControl::reconnectTo`  
[protected]

**6.217.3.8** `bool activemq::commands::ConnectionControl::resume`  
[protected]

**6.217.3.9** `bool activemq::commands::ConnectionControl::suspend`  
[protected]

The documentation for this class was generated from the following file:

6.218

**activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller**

**Class Reference**

1307

- [src/main/activemq/commands/ConnectionControl.h](#)

## 6.218 **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller** **Class Reference**

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1307).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller**:

### Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.218.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1307).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.218.2 Constructor & Destructor Documentation

6.218.2.1 **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::ConnectionControlMarshaller**  
 ( ) [inline]

6.218.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::~~ConnectionControlMarshaller**  
 ( ) [inline, virtual]

### 6.218.3 Member Function Documentation

6.218.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.218.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::getDataStructure**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.218.3.3 **virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::looseMarshal**  
 ( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

6.218

**activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller**

**Class Reference**

**1309**

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 772).

**6.218.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 774).

**6.218.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 775).

```
6.218.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 776).

```
6.218.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## 6.219

**activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller**

**Class Reference**

**1311**

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionControlMarshaller.h**

## 6.219 **activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1311).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionControlMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller**:

### Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.219.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1311).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.219.2 Constructor & Destructor Documentation

6.219.2.1 **activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::ConnectionControlMarshaller**  
 ( ) [inline]

6.219.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::~~ConnectionControlMarshaller**  
 ( ) [inline, virtual]

### 6.219.3 Member Function Documentation

6.219.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.219.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::getDataStructure**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value



## 6.219

**activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller**

### Class Reference

1313

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.219.3.3** virtual void **activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::looseMarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 780).

**6.219.3.4** virtual void **activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 781).

```

6.219.3.5  virtual int activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 782).

```

6.219.3.6  virtual void activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 783).

6.220

**activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller**

**Class Reference**

1315

6.219.3.7 **virtual void activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::tightUnmarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream**  
\* *bs* ) **throw ( decaf::io::IOException )** [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ConnectionControlMarshaller.h**

## 6.220 **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1315).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller**:

#### Public Member Functions

- **ConnectionControlMarshaller ()**
- virtual **~ConnectionControlMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.220.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1315).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.220.2 Constructor & Destructor Documentation

6.220.2.1 **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::ConnectionControlMarshaller**  
( ) [inline]

6.220.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::~~ConnectionControlMarshaller**  
( ) [inline, virtual]

### 6.220.3 Member Function Documentation

6.220.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

6.220

**activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller**

**Class Reference**

**1317**

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.220.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.220.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 787).

**6.220.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 788).

```
6.220.3.5  virtual int activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 789).

```
6.220.3.6  virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

## 6.221

**activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller**

**Class Reference**

**1319**

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 790).

**6.220.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 792).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h`

## 6.221 **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1319).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller**:

## Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.221.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1319).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.221

**activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller**

Class Reference

1321

### 6.221.2 Constructor & Destructor Documentation

6.221.2.1 **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::ConnectionControlMarshaller**  
( ) [inline]

6.221.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::~~ConnectionControlMarshaller**  
( ) [inline, virtual]

### 6.221.3 Member Function Documentation

6.221.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.221.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.221.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 794).

**6.221.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 795).

**6.221.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.221

**activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller**

### Class Reference

1323

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 796).

```
6.221.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 797).

```
6.221.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h`

## 6.222 `activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1324).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ConnectionControlMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller`:

### Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

---

*Write a object instance to data output stream.*

### 6.222.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1324).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.222.2 Constructor & Destructor Documentation

6.222.2.1 **activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::ConnectionControlMarshaller**  
( ) [inline]

6.222.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::~~ConnectionControlMarshaller**  
( ) [inline, virtual]

### 6.222.3 Member Function Documentation

6.222.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.222.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.222.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 801).

**6.222.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 802).

**6.222.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## 6.222

activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller

## Class Reference

1327

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 803).

**6.222.3.6** virtual void activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::tightMarshal2 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 804).

**6.222.3.7** virtual void activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 806).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ConnectionControlMarshaller.h`

## 6.223 **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1328).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller**:

**Public Member Functions**

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*



## 6.223

**activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller**

### Class Reference

1329

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.223.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1328).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.223.2 Constructor & Destructor Documentation

6.223.2.1 **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::ConnectionControlMarshaller**  
( ) [inline]

6.223.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::~~ConnectionControlMarshaller**  
( ) [inline, virtual]

### 6.223.3 Member Function Documentation

6.223.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.223.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

```
6.223.3.3  virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 808).

```
6.223.3.4  virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 809).

## 6.223

**activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller**

### Class Reference

1331

6.223.3.5 **virtual int activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::tightMarshal1**  
**( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )**  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 810).

6.223.3.6 **virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::tightMarshal2**  
**( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
 utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )**  
 [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 811).

```
6.223.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 813).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h

## 6.224 activemq::commands::ConnectionError Class Reference

```
#include <src/main/activemq/commands/ConnectionError.h>
```

Inheritance diagram for **activemq::commands::ConnectionError**:

#### Public Member Functions

- **ConnectionError** ()
- virtual **~ConnectionError** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ConnectionError \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*

- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataSet** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataSet** \*value) const  
*Compares the **DataSet** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **BrokerError** > & **getException** () const
- virtual **Pointer**< **BrokerError** > & **getException** ()
- virtual void **setException** (const **Pointer**< **BrokerError** > &exception)
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_CONNECTIONERROR** = 16

### Protected Attributes

- **Pointer**< **BrokerError** > exception
- **Pointer**< **ConnectionId** > connectionId

## 6.224.1 Constructor & Destructor Documentation

6.224.1.1 **activemq::commands::ConnectionError::ConnectionError** ( )

6.224.1.2 **virtual activemq::commands::ConnectionError::~~ConnectionError** ( )  
[virtual]

## 6.224.2 Member Function Documentation

6.224.2.1 **virtual ConnectionError\*** **activemq::commands::ConnectionError::cloneDataSet**  
( )const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

**Returns**

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

**6.224.2.2** `virtual void activemq::commands::ConnectionError::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

**Parameters**

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

**6.224.2.3** `virtual bool activemq::commands::ConnectionError::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

**6.224.2.4** `virtual const Pointer<ConnectionId>& activemq::commands::ConnectionError::getConnectionId ( ) const [virtual]`

**6.224.2.5** `virtual Pointer<ConnectionId>& activemq::commands::ConnectionError::getConnectionId ( ) [virtual]`

**6.224.2.6** `virtual unsigned char activemq::commands::ConnectionError::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns**

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

- 6.224.2.7 `virtual Pointer<BrokerError>& activemq::commands::ConnectionError::getException ( )`  
[virtual]
- 6.224.2.8 `virtual const Pointer<BrokerError>& activemq::commands::ConnectionError::getException ( ) const`  
[virtual]
- 6.224.2.9 `virtual void activemq::commands::ConnectionError::setConnectionId ( const Pointer< ConnectionId > & connectionId )` [virtual]
- 6.224.2.10 `virtual void activemq::commands::ConnectionError::setException ( const Pointer< BrokerError > & exception )` [virtual]
- 6.224.2.11 `virtual std::string activemq::commands::ConnectionError::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataSet** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

- 6.224.2.12 `virtual Pointer<Command> activemq::commands::ConnectionError::visit ( activemq::state::CommandVisitor * visitor ) throw ( exceptions::ActiveMQException )` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

### 6.224.3 Field Documentation

6.224.3.1 **Pointer<ConnectionId> activemq::commands::ConnectionError::connectionId**  
[protected]

6.224.3.2 **Pointer<BrokerError> activemq::commands::ConnectionError::exception**  
[protected]

6.224.3.3 **const unsigned char activemq::commands::ConnectionError::ID\_-  
CONNECTIONERROR = 16** [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ConnectionError.h**

## 6.225 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1336).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller:

### Public Member Functions

- **ConnectionErrorMarshaller ()**
- virtual **~ConnectionErrorMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)**  
throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*



## 6.225

**activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller**

### Class Reference

1337

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

## 6.225.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1336).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.225.2 Constructor & Destructor Documentation

6.225.2.1 **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::ConnectionErrorMarshaller**  
( ) [inline]

6.225.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller**  
( ) [inline, virtual]

## 6.225.3 Member Function Documentation

6.225.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.225.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.225.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 808).

**6.225.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 809).

6.225

activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller

Class Reference

1339

6.225.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::tightMarshal1 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 810).

6.225.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::tightMarshal2 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 811).

```
6.225.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 813).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h

## 6.226 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1340).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller**:

#### Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*

## 6.226

**activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller**

### Class Reference

1341

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.226.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1340).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.226.2 Constructor & Destructor Documentation

6.226.2.1 **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::ConnectionErrorMarshaller**  
( ) [inline]

6.226.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller**  
( ) [inline, virtual]

### 6.226.3 Member Function Documentation

6.226.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.226.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.226.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 772).

**6.226.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

## 6.226

**activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller**

### Class Reference

1343

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 774).

```
6.226.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 775).

```
6.226.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 776).

```
6.226.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h

## 6.227 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1344).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller**:



- **ConnectionErrorMarshaller ()**
- virtual **~ConnectionErrorMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.227.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1344).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.227.2 Constructor & Destructor Documentation

6.227.2.1 `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::ConnectionErrorMarshaller ( ) [inline]`

6.227.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller ( ) [inline, virtual]`

## 6.227.3 Member Function Documentation

6.227.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.227.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.227.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

6.227

**activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller****Class Reference****1347****Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 780).

**6.227.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 781).

**6.227.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 782).

```
6.227.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 783).

```
6.227.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 784).

The documentation for this class was generated from the following file:

6.228

**activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller**

**Class Reference**

1349

- `src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h`

## 6.228 **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** **Class Reference**

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1349).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller`:

### Public Member Functions

- **ConnectionErrorMarshaller ()**
- virtual **~ConnectionErrorMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**

*Write a object instance to data output stream.*

### 6.228.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1349).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.228.2 Constructor & Destructor Documentation

6.228.2.1 **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::ConnectionErrorMarshaller**  
 ( ) [inline]

6.228.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller**  
 ( ) [inline, virtual]

### 6.228.3 Member Function Documentation

6.228.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::createObject (**  
**) const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.228.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::getDataStructureType**  
**( ) const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.228.3.3 **virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure**  
**\* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw (**  
**decaf::io::IOException)** [virtual]

Write a object instance to data output stream.

6.228

**activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller**

**Class Reference**

**1351**

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 787).

**6.228.3.4** virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::looseUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 788).

**6.228.3.5** virtual int activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::tightMarshal1 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 789).

```
6.228.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 790).

```
6.228.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.



6.229

**activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller**

**Class Reference**

**1353**

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 792).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ConnectionErrorMarshaller.h**

## 6.229 **activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1353).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller**:

### Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.229.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1353).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.229.2 Constructor & Destructor Documentation

- 6.229.2.1 **activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::ConnectionErrorMarshaller**  
 ( ) [inline]
- 6.229.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller**  
 ( ) [inline, virtual]

### 6.229.3 Member Function Documentation

- 6.229.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

- 6.229.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

## 6.229

**activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller**

**Class Reference**

**1355**

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.229.3.3** virtual void **activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::looseMarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 794).

**6.229.3.4** virtual void **activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 795).

```
6.229.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 796).

```
6.229.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 797).

## 6.230

**activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller**

**Class Reference**

1357

6.229.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h`

## 6.230 **activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1357).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ConnectionErrorMarshaller>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller**:

### Public Member Functions

- **ConnectionErrorMarshaller ()**
- virtual **~ConnectionErrorMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.230.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1357).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.230.2 Constructor & Destructor Documentation

6.230.2.1 **activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::ConnectionErrorMarshaller**  
( ) [inline]

6.230.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller**  
( ) [inline, virtual]

### 6.230.3 Member Function Documentation

6.230.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

## 6.230

**activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller**

**Class Reference**

**1359**

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.230.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.230.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 801).

**6.230.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 802).

```
6.230.3.5  virtual int activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 803).

```
6.230.3.6  virtual void activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled



<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 804).

**6.230.3.7** `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 806).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ConnectionErrorMarshaller.h`

**6.231 cms::ConnectionFactory Class Reference**

Defines the interface for a factory that creates connection objects, the **Connection** (p. 1296) objects returned implement the CMS **Connection** (p. 1296) interface and hide the CMS Provider specific implementation details behind that interface.

```
#include <src/main/cms/ConnectionFactory.h>
```

Inheritance diagram for cms::ConnectionFactory:

## Public Member Functions

- virtual `~ConnectionFactory()`
- virtual `Connection * createConnection()=0` throw ( `CMSEException` )  
*Creates a connection with the default user identity.*
- virtual `cms::Connection * createConnection` (const std::string &username, const std::string &password)=0 throw ( `cms::CMSEException` )  
*Creates a connection with the default specified identity.*
- virtual `cms::Connection * createConnection` (const std::string &username, const std::string &password, const std::string &clientId)=0 throw ( `cms::CMSEException` )  
*Creates a connection with the specified user identity.*

## Static Public Member Functions

- static `ConnectionFactory * createCMSConnectionFactory` (const std::string &brokerURI) throw ( `cms::CMSEException` )  
*Static method that is used to create a provider specific connection factory.*

### 6.231.1 Detailed Description

Defines the interface for a factory that creates connection objects, the **Connection** (p. 1296) objects returned implement the CMS **Connection** (p. 1296) interface and hide the CMS Provider specific implementation details behind that interface. A Client creates a new **ConnectionFactory** (p. 1361) either directly by instantiating the provider specific implementation of the factory or by using the static method `createCMSConnectionFactory` which all providers are required to implement.

#### Since

1.0

### 6.231.2 Constructor & Destructor Documentation

- 6.231.2.1 `virtual cms::ConnectionFactory::~~ConnectionFactory ( ) [inline, virtual]`

### 6.231.3 Member Function Documentation

- 6.231.3.1 `static ConnectionFactory* cms::ConnectionFactory::createCMSConnectionFactory ( const std::string & brokerURI ) throw ( cms::CMSEException ) [static]`

Static method that is used to create a provider specific connection factory.

The provider implements this method in their library and returns an instance of a **ConnectionFactory** (p. 1361) derived object. Clients can use this method to remain abstracted from the specific CMS implementation being used.

#### Parameters

<i>brokerURI</i>	The remote address to use to connect to the Provider.
------------------	---

#### Returns

A pointer to a provider specific implementation of the **ConnectionFactory** (p. 1361) interface, the caller is responsible for deleting this resource.

#### Exceptions

<b>CMSException</b> (p. 1190)	if an internal error occurs while creating the <b>ConnectionFactory</b> (p. 1361).
----------------------------------	--

**6.231.3.2** `virtual cms::Connection* cms::ConnectionFactory::createConnection  
( const std::string & username, const std::string & password ) throw (  
cms::CMSException ) [pure virtual]`

Creates a connection with the default specified identity.

The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 3692) method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless `createConnection` will continue to use the default values that were set in the Constructor.

#### Parameters

<i>username</i>	The user name used to authenticate with the Provider.
<i>password</i>	The password used to authenticate with the Provider.

#### Returns

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

#### Exceptions

<b>CMSException</b> (p. 1190)	if an internal error occurs while creating the <b>Connection</b> (p. 1296).
----------------------------------	---

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 285).

**6.231.3.3** `virtual cms::Connection* cms::ConnectionFactory::createConnection ( const std::string & username, const std::string & password, const std::string & clientId ) throw ( cms::CMSEException ) [pure virtual]`

Creates a connection with the specified user identity.

The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 3692) method is explicitly called. The user name and password values passed here do not change the defaults, subsequent calls to the parameterless `createConnection` will continue to use the default values that were set in the Constructor.

#### Parameters

<i>username</i>	The user name used to authenticate with the Provider.
<i>password</i>	The password used to authenticate with the Provider.
<i>clientId</i>	The Client Id assigned to connection. If the id is the empty string ("" ) then a random client Id is created for this connection.

#### Returns

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

#### Exceptions

<b>CMSEException</b> (p. 1190)	if an internal error occurs while creating the <b>Connection</b> (p. 1296).
-----------------------------------	---

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 286).

**6.231.3.4** `virtual Connection* cms::ConnectionFactory::createConnection ( ) throw ( CMSEException ) [pure virtual]`

Creates a connection with the default user identity.

The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 3692) method is explicitly called.

#### Returns

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

#### Exceptions

<b>CMSEException</b> (p. 1190)	if an internal error occurs while creating the <b>Connection</b> (p. 1296).
-----------------------------------	---

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 285).

The documentation for this class was generated from the following file:

- src/main/cms/ConnectionFactory.h

## 6.232 activemq::commands::ConnectionId Class Reference

```
#include <src/main/activemq/commands/ConnectionId.h>
```

Inheritance diagram for activemq::commands::ConnectionId:

### Public Types

- typedef **decaf::lang::PointerComparator**< **ConnectionId** > **COMPARATOR**

### Public Member Functions

- **ConnectionId** ()
- **ConnectionId** (const **ConnectionId** &other)
- **ConnectionId** (const **SessionId** \*sessionId)
- **ConnectionId** (const **ProducerId** \*producerId)
- **ConnectionId** (const **ConsumerId** \*consumerId)
- virtual ~**ConnectionId** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ConnectionId** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getValue** () const
- virtual std::string & **getValue** ()
- virtual void **setValue** (const std::string &value)
- virtual int **compareTo** (const **ConnectionId** &value) const

- virtual bool **equals** (const **ConnectionId** &value) const
- virtual bool **operator==** (const **ConnectionId** &value) const
- virtual bool **operator<** (const **ConnectionId** &value) const
- **ConnectionId** & **operator=** (const **ConnectionId** &other)

### Static Public Attributes

- static const unsigned char **ID\_CONNECTIONID** = 120

### Protected Attributes

- std::string value

## 6.232.1 Member Typedef Documentation

6.232.1.1 **typedef** decaf::lang::PointerComparator<ConnectionId>  
**activemq::commands::ConnectionId::COMPARATOR**

## 6.232.2 Constructor & Destructor Documentation

6.232.2.1 **activemq::commands::ConnectionId::ConnectionId** ( )

6.232.2.2 **activemq::commands::ConnectionId::ConnectionId** ( const **ConnectionId** & *other* )

6.232.2.3 **activemq::commands::ConnectionId::ConnectionId** ( const **SessionId** \* *sessionId* )

6.232.2.4 **activemq::commands::ConnectionId::ConnectionId** ( const **ProducerId** \*  
*producerId* )

6.232.2.5 **activemq::commands::ConnectionId::ConnectionId** ( const **ConsumerId** \*  
*consumerId* )

6.232.2.6 **virtual** **activemq::commands::ConnectionId::~~ConnectionId** ( ) [virtual]

## 6.232.3 Member Function Documentation

6.232.3.1 **virtual** **ConnectionId\*** **activemq::commands::ConnectionId::cloneDataStructure** ( )  
**const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

**6.232.3.2** `virtual int activemq::commands::ConnectionId::compareTo ( const ConnectionId & value ) const` [virtual]

**6.232.3.3** `virtual void activemq::commands::ConnectionId::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1715).

**6.232.3.4** `virtual bool activemq::commands::ConnectionId::equals ( const DataStructure * value ) const` [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1716).

**6.232.3.5** `virtual bool activemq::commands::ConnectionId::equals ( const ConnectionId & value ) const` [virtual]

**6.232.3.6** `virtual unsigned char activemq::commands::ConnectionId::getDataStructureType ( ) const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

- 6.232.3.7 `virtual const std::string& activemq::commands::ConnectionId::getValue ( ) const`  
[virtual]
- 6.232.3.8 `virtual std::string& activemq::commands::ConnectionId::getValue ( )`  
[virtual]
- 6.232.3.9 `virtual bool activemq::commands::ConnectionId::operator< ( const ConnectionId  
& value ) const` [virtual]
- 6.232.3.10 `ConnectionId& activemq::commands::ConnectionId::operator= ( const  
ConnectionId & other )`
- 6.232.3.11 `virtual bool activemq::commands::ConnectionId::operator== ( const ConnectionId  
& value ) const` [virtual]
- 6.232.3.12 `virtual void activemq::commands::ConnectionId::setValue ( const std::string & value  
)` [virtual]
- 6.232.3.13 `virtual std::string activemq::commands::ConnectionId::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 841).

### 6.232.4 Field Documentation

- 6.232.4.1 `const unsigned char activemq::commands::ConnectionId::ID_-  
CONNECTIONID = 120` [static]

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

- 6.232.4.2 `std::string activemq::commands::ConnectionId::value` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionId.h`

## 6.233 **activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1368).



```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller:

## Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.233.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1368). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.233.2 Constructor & Destructor Documentation

6.233.2.1 `activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::ConnectionIdMarshaller ( ) [inline]`

6.233.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::~~ConnectionIdMarshaller ( ) [inline, virtual]`

## 6.233.3 Member Function Documentation

6.233.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.233.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.233.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.233.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.233.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```

6.233.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```

6.233.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h

## 6.234 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1373).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller:

### Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.234.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1373). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.234.2 Constructor & Destructor Documentation

6.234.2.1 **activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::ConnectionIdMarshaller**  
( ) [inline]

6.234.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::~~ConnectionIdMarshaller**  
( ) [inline, virtual]

### 6.234.3 Member Function Documentation

6.234.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.234.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.234.3.3 **virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.234.3.4** virtual void **activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.234.3.5** virtual int **activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::tightMarshal1** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.234.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.234.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:



- src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h

## 6.235 activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1377).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller:

### Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.235.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1377). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.235.2 Constructor & Destructor Documentation

6.235.2.1 **activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::ConnectionIdMarshaller**  
( ) [inline]

6.235.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::~~ConnectionIdMarshaller**  
( ) [inline, virtual]

### 6.235.3 Member Function Documentation

6.235.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.235.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.235.3.3 **virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.235.3.4** virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::looseUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.235.3.5** virtual int activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::tightMarshal1 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.235.3.6  virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.235.3.7  virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h`

## 6.236 **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1381).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller**:

### Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.236.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1381). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.236.2 Constructor & Destructor Documentation

6.236.2.1 **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::ConnectionIdMarshaller**  
( ) [inline]

6.236.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::~~ConnectionIdMarshaller**  
( ) [inline, virtual]

### 6.236.3 Member Function Documentation

6.236.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.236.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.236.3.3 **virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.236.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.236.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.236.3.6  virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.236.3.7  virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:



- src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h

## 6.237 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1385).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller:

### Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.237.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1385). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.237.2 Constructor & Destructor Documentation

6.237.2.1 **activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::ConnectionIdMarshaller**  
( ) [inline]

6.237.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::~~ConnectionIdMarshaller**  
( ) [inline, virtual]

### 6.237.3 Member Function Documentation

6.237.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.237.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.237.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.237.3.4** virtual void activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::looseUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.237.3.5** virtual int activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::tightMarshal1 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.237.3.6  virtual void activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.237.3.7  virtual void activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h

## 6.238 activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1389).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ConnectionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller:

### Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.238.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1389). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.238.2 Constructor & Destructor Documentation

6.238.2.1 **activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::ConnectionIdMarshaller**  
( ) [inline]

6.238.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::~~ConnectionIdMarshaller**  
( ) [inline, virtual]

### 6.238.3 Member Function Documentation

6.238.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.238.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.238.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.238.3.4** virtual void activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::looseUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.238.3.5** virtual int activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::tightMarshal1 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.238.3.6  virtual void activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.238.3.7  virtual void activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:



- src/main/activemq/wireformat/openwire/marshall/v6/ConnectionIdMarshaller.h

## 6.239 activemq::commands::ConnectionInfo Class Reference

```
#include <src/main/activemq/commands/ConnectionInfo.h>
```

Inheritance diagram for activemq::commands::ConnectionInfo:

### Public Member Functions

- **ConnectionInfo** ()
- virtual **~ConnectionInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ConnectionInfo** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- **Pointer**< **RemoveInfo** > **createRemoveCommand** () const
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual const std::string & **getPassword** () const
- virtual std::string & **getPassword** ()
- virtual void **setPassword** (const std::string &password)
- virtual const std::string & **getUserName** () const
- virtual std::string & **getUserName** ()
- virtual void **setUserName** (const std::string &userName)

- virtual const std::vector< **decaf::lang::Pointer< BrokerId > >** & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer< BrokerId > >** & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer< BrokerId > >** & **brokerPath**)
- virtual bool **isBrokerMasterConnector** () const
- virtual void **setBrokerMasterConnector** (bool **brokerMasterConnector**)
- virtual bool **isManageable** () const
- virtual void **setManageable** (bool **manageable**)
- virtual bool **isClientMaster** () const
- virtual void **setClientMaster** (bool **clientMaster**)
- virtual bool **isFaultTolerant** () const
- virtual void **setFaultTolerant** (bool **faultTolerant**)
- virtual bool **isConnectionInfo** () const
- virtual **Pointer< Command >** **visit** (**activemq::state::CommandVisitor** \*visitor) throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_CONNECTIONINFO** = 3

### Protected Attributes

- **Pointer< ConnectionId >** **connectionId**
- std::string **clientId**
- std::string **password**
- std::string **userName**
- std::vector< **decaf::lang::Pointer< BrokerId > >** **brokerPath**
- bool **brokerMasterConnector**
- bool **manageable**
- bool **clientMaster**
- bool **faultTolerant**

### 6.239.1 Constructor & Destructor Documentation

6.239.1.1 `activemq::commands::ConnectionInfo::ConnectionInfo ( )`

6.239.1.2 `virtual activemq::commands::ConnectionInfo::~~ConnectionInfo ( ) [virtual]`

### 6.239.2 Member Function Documentation

6.239.2.1 `virtual ConnectionInfo* activemq::commands::ConnectionInfo::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1714).

6.239.2.2 `virtual void activemq::commands::ConnectionInfo::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

6.239.2.3 `Pointer<RemoveInfo> activemq::commands::ConnectionInfo::createRemoveCommand ( ) const`

6.239.2.4 `virtual bool activemq::commands::ConnectionInfo::equals ( const DataStructure * value ) const [virtual]`

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

- 6.239.2.5 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConnectionInfo::getBrokerPath ( ) const [virtual]`
- 6.239.2.6 `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConnectionInfo::getBrokerPath ( ) [virtual]`
- 6.239.2.7 `virtual const std::string& activemq::commands::ConnectionInfo::getClientId ( ) const [virtual]`
- 6.239.2.8 `virtual std::string& activemq::commands::ConnectionInfo::getClientId ( ) [virtual]`
- 6.239.2.9 `virtual Pointer<ConnectionId>& activemq::commands::ConnectionInfo::getConnectionId ( ) [virtual]`
- 6.239.2.10 `virtual const Pointer<ConnectionId>& activemq::commands::ConnectionInfo::getConnectionId ( ) const [virtual]`
- 6.239.2.11 `virtual unsigned char activemq::commands::ConnectionInfo::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

### Returns

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

6.239.2.12 `virtual const std::string& activemq::commands::ConnectionInfo::getPassword ( )`  
`const [virtual]`

6.239.2.13 `virtual std::string& activemq::commands::ConnectionInfo::getPassword ( )`  
`[virtual]`

6.239.2.14 `virtual const std::string& activemq::commands::ConnectionInfo::getUserName ( )`  
`const [virtual]`

6.239.2.15 `virtual std::string& activemq::commands::ConnectionInfo::getUserName ( )`  
`[virtual]`

6.239.2.16 `virtual bool activemq::commands::ConnectionInfo::isBrokerMasterConnector ( )`  
`const [virtual]`

6.239.2.17 `virtual bool activemq::commands::ConnectionInfo::isClientMaster ( ) const`  
`[virtual]`

6.239.2.18 `virtual bool activemq::commands::ConnectionInfo::isConnectionInfo ( ) const`  
`[inline, virtual]`

### Returns

an answer of true to the `isConnectionInfo()` (p. 1397) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 768).

- 6.239.2.19 `virtual bool activemq::commands::ConnectionInfo::isFaultTolerant ( ) const`  
[virtual]
- 6.239.2.20 `virtual bool activemq::commands::ConnectionInfo::isManageable ( ) const`  
[virtual]
- 6.239.2.21 `virtual void activemq::commands::ConnectionInfo::setBrokerMasterConnector ( bool  
brokerMasterConnector )` [virtual]
- 6.239.2.22 `virtual void activemq::commands::ConnectionInfo::setBrokerPath ( const  
std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath )`  
[virtual]
- 6.239.2.23 `virtual void activemq::commands::ConnectionInfo::setClientId ( const std::string &  
clientId )` [virtual]
- 6.239.2.24 `virtual void activemq::commands::ConnectionInfo::setClientMaster ( bool  
clientMaster )` [virtual]
- 6.239.2.25 `virtual void activemq::commands::ConnectionInfo::setConnectionId ( const  
Pointer< ConnectionId > & connectionId )` [virtual]
- 6.239.2.26 `virtual void activemq::commands::ConnectionInfo::setFaultTolerant ( bool  
faultTolerant )` [virtual]
- 6.239.2.27 `virtual void activemq::commands::ConnectionInfo::setManageable ( bool  
manageable )` [virtual]
- 6.239.2.28 `virtual void activemq::commands::ConnectionInfo::setPassword ( const std::string &  
password )` [virtual]
- 6.239.2.29 `virtual void activemq::commands::ConnectionInfo::setUserName ( const std::string  
& userName )` [virtual]
- 6.239.2.30 `virtual std::string activemq::commands::ConnectionInfo::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

**6.239.2.31** `virtual Pointer<Command> activemq::commands::ConnectionInfo::visit  
( activemq::state::CommandVisitor * visitor ) throw (  
exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

### 6.239.3 Field Documentation

**6.239.3.1** `bool activemq::commands::ConnectionInfo::brokerMasterConnector  
[protected]`

**6.239.3.2** `std::vector< decaf::lang::Pointer<BrokerId> >  
activemq::commands::ConnectionInfo::brokerPath [protected]`

**6.239.3.3** `std::string activemq::commands::ConnectionInfo::clientId  
[protected]`

**6.239.3.4** `bool activemq::commands::ConnectionInfo::clientMaster  
[protected]`

**6.239.3.5** `Pointer<ConnectionId> activemq::commands::ConnectionInfo::connectionId  
[protected]`

**6.239.3.6** `bool activemq::commands::ConnectionInfo::faultTolerant  
[protected]`

**6.239.3.7** `const unsigned char activemq::commands::ConnectionInfo::ID_-  
CONNECTIONINFO = 3 [static]`

**6.239.3.8** `bool activemq::commands::ConnectionInfo::manageable  
[protected]`

**6.239.3.9** `std::string activemq::commands::ConnectionInfo::password  
[protected]`

**6.239.3.10** `std::string activemq::commands::ConnectionInfo::userName  
[protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionInfo.h`

## 6.240 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1400).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller:

### Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*



### 6.240.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1400).  
NOTE!: This file is auto generated - do not modify! if you need to make a change,  
please see the Java Classes in the activemq-openwire-generator module

### 6.240.2 Constructor & Destructor Documentation

**6.240.2.1** `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::ConnectionInfoMarshaller  
( ) [inline]`

**6.240.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::~~ConnectionInfoMarshaller  
( ) [inline, virtual]`

### 6.240.3 Member Function Documentation

**6.240.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::createObject (  
) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.240.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::getDataStructureType  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.240.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::looseMarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (  
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 808).

**6.240.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 809).

**6.240.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 810).

**6.240.3.6** virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::tightMarshal2  
( OpenWireFormat \* *wireFormat*, commands::DataStructure  
\* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 811).

**6.240.3.7** virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::tightUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream  
\* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 813).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h`

## 6.241 **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1404).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller**:

**Public Member Functions**

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.241.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1404).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.241.2 Constructor & Destructor Documentation

6.241.2.1 **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::ConnectionInfoMarshaller**  
( ) [inline]

6.241.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::~~ConnectionInfoMarshaller**  
( ) [inline, virtual]

### 6.241.3 Member Function Documentation

6.241.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.241.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.241.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 772).

**6.241.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 774).

6.241.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::tightMarshal1 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 775).

6.241.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::tightMarshal2 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 776).

```
6.241.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h

## 6.242 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1408).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller**:

#### Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*



- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.242.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1408).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.242.2 Constructor & Destructor Documentation

6.242.2.1 **activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::ConnectionInfoMarshaller**  
( ) [inline]

6.242.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::~~ConnectionInfoMarshaller**  
( ) [inline, virtual]

### 6.242.3 Member Function Documentation

6.242.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.242.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.242.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 780).

**6.242.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 781).

**6.242.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 782).

**6.242.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 783).

**6.242.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h`

## 6.243 **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1412).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller**:

## Public Member Functions

- **ConnectionInfoMarshaller ()**
- virtual **~ConnectionInfoMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.243.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1412).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.243.2 Constructor & Destructor Documentation

6.243.2.1 `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::ConnectionInfoMarshaller ( ) [inline]`

6.243.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::~~ConnectionInfoMarshaller ( ) [inline, virtual]`

## 6.243.3 Member Function Documentation

6.243.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.243.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.243.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 787).

**6.243.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 788).

**6.243.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 789).

```
6.243.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 790).

```
6.243.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 792).

The documentation for this class was generated from the following file:



- src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h

## 6.244 activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1417).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller:

### Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.244.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1417).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.244.2 Constructor & Destructor Documentation

6.244.2.1 **activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::ConnectionInfoMarshaller**  
 ( ) [inline]

6.244.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::~~ConnectionInfoMarshaller**  
 ( ) [inline, virtual]

### 6.244.3 Member Function Documentation

6.244.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::createObject** ( ) **const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.244.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::getDataStructureType**  
 ( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.244.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::looseMarshal**  
 ( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 794).

**6.244.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 795).

**6.244.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 796).

```
6.244.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 797).

```
6.244.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## 6.245 activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller Class Reference 1421

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h

## 6.245 activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1421).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ConnectionInfoMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller**:

### Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.245.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1421).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.245.2 Constructor & Destructor Documentation

6.245.2.1 **activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::ConnectionInfoMarshaller**  
 ( ) [inline]

6.245.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::~~ConnectionInfoMarshaller**  
 ( ) [inline, virtual]

### 6.245.3 Member Function Documentation

6.245.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.245.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

## 6.245 activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller

### Class Reference 1423

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.245.3.3** virtual void **activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::looseMarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 801).

**6.245.3.4** virtual void **activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 802).

```

6.245.3.5  virtual int activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 803).

```

6.245.3.6  virtual void activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 804).



6.245.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 806).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ConnectionInfoMarshaller.h**

## 6.246 cms::ConnectionMetaData Class Reference

A **ConnectionMetaData** (p. 1425) object provides information describing the **Connection** (p. 1296) object.

```
#include <src/main/cms/ConnectionMetaData.h>
```

Inheritance diagram for cms::ConnectionMetaData:

#### Public Member Functions

- virtual ~**ConnectionMetaData** ()
- virtual std::string **getCMSVersion** () const =0 throw ( cms::CMSEException )  
*Gets the CMS API version.*
- virtual int **getCMSMajorVersion** () const =0 throw ( cms::CMSEException )  
*Gets the CMS major version number.*
- virtual int **getCMSMinorVersion** () const =0 throw ( cms::CMSEException )  
*Gets the CMS minor version number.*

- virtual std::string **getCMSProviderName** () const =0 throw ( cms::CMSException )  
*Gets the CMS provider name.*
- virtual std::string **getProviderVersion** () const =0 throw ( cms::CMSException )  
*Gets the CMS provider version.*
- virtual int **getProviderMajorVersion** () const =0 throw ( cms::CMSException )  
*Gets the CMS provider major version number.*
- virtual int **getProviderMinorVersion** () const =0 throw ( cms::CMSException )  
*Gets the CMS provider minor version number.*
- virtual std::vector< std::string > **getCMSXPropertyNames** () const =0 throw ( cms::CMSException )  
*Gets an Vector of the CMSX property names.*

### 6.246.1 Detailed Description

A **ConnectionMetaData** (p. 1425) object provides information describing the **Connection** (p. 1296) object.

Since

1.3

### 6.246.2 Constructor & Destructor Documentation

- 6.246.2.1 virtual cms::ConnectionMetaData::~~ConnectionMetaData ( ) [inline, virtual]

### 6.246.3 Member Function Documentation

- 6.246.3.1 virtual int cms::ConnectionMetaData::getCMSMajorVersion ( ) const throw ( cms::CMSException ) [pure virtual]

Gets the CMS major version number.

Returns

the CMS API major version number

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	If the CMS Provider fails to retrieve the metadata due to some internal error.
---	--

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 294).

**6.246.3.2** `virtual int cms::ConnectionMetaData::getCMSMinorVersion ( ) const throw ( cms::CMSException ) [pure virtual]`

Gets the CMS minor version number.

**Returns**

the CMS API minor version number

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	If the CMS Provider fails to retrieve the metadata due to some internal error.
---	--

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 294).

**6.246.3.3** `virtual std::string cms::ConnectionMetaData::getCMSProviderName ( ) const throw ( cms::CMSException ) [pure virtual]`

Gets the CMS provider name.

**Returns**

the CMS provider name

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	If the CMS Provider fails to retrieve the metadata due to some internal error.
---	--

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 295).

**6.246.3.4** `virtual std::string cms::ConnectionMetaData::getCMSVersion ( ) const throw ( cms::CMSException ) [pure virtual]`

Gets the CMS API version.

**Returns**

the CMS API Version in String form.

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	If the CMS Provider fails to retrieve the metadata due to some internal error.
---	--

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 295).

**6.246.3.5** `virtual std::vector<std::string> cms::ConnectionMetaData::getCMSXPropertyNames ( ) const throw ( cms::CMSException ) [pure virtual]`

Gets an Vector of the CMSX property names.

**Returns**

an Vector of CMSX property names

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	If the CMS Provider fails to retrieve the metadata due to some internal error.
---	--

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 295).

**6.246.3.6** `virtual int cms::ConnectionMetaData::getProviderMajorVersion ( ) const throw ( cms::CMSException ) [pure virtual]`

Gets the CMS provider major version number.

**Returns**

the CMS provider major version number

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	If the CMS Provider fails to retrieve the metadata due to some internal error.
---	--

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 296).

**6.246.3.7** `virtual int cms::ConnectionMetaData::getProviderMinorVersion ( ) const throw ( cms::CMSException ) [pure virtual]`

Gets the CMS provider minor version number.

**Returns**

the CMS provider minor version number

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	If the CMS Provider fails to retrieve the metadata due to some internal error.
---	--

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 296).

**6.246.3.8** `virtual std::string cms::ConnectionMetaData::getProviderVersion ( ) const throw ( cms::CMSException ) [pure virtual]`

Gets the CMS provider version.

**Returns**

the CMS provider version

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	If the CMS Provider fails to retrieve the metadata due to some internal error.
---	--

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 296).

The documentation for this class was generated from the following file:

- `src/main/cms/ConnectionMetaData.h`

**6.247 activemq::state::ConnectionState Class Reference**

```
#include <src/main/activemq/state/ConnectionState.h>
```

**Public Member Functions**

- **ConnectionState** (const **Pointer**< **ConnectionInfo** > &info)
- virtual **~ConnectionState** ()
- `std::string toString () const`
- const **Pointer**< **commands::ConnectionInfo** > & **getInfo** () const
- void **checkShutdown** () const
- void **shutdown** ()
- void **reset** (const **Pointer**< **ConnectionInfo** > &info)
- void **addTempDestination** (const **Pointer**< **DestinationInfo** > &info)
- void **removeTempDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- void **addTransactionState** (const **Pointer**< **TransactionId** > &id)
- const **Pointer**< **TransactionState** > & **getTransactionState** (const **Pointer**< **TransactionId** > &id) const
- `std::vector< Pointer< TransactionState > > getTransactionStates () const`

- **Pointer< TransactionState > removeTransactionState** (const **Pointer< TransactionId > &id**)
- **void addSession** (const **Pointer< SessionInfo > &info**)
- **Pointer< SessionState > removeSession** (const **Pointer< SessionId > &id**)
- **const Pointer< SessionState > & getSessionState** (const **Pointer< SessionId > &id**) const
- **const StlList< Pointer< DestinationInfo > > & getTempDesinations** () const
- **std::vector< Pointer< SessionState > > getSessionStates** () const
- **StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > getRecoveringPullConsumers** ()
- **void setConnectionInterruptProcessingComplete** (bool connectionInterruptProcessingComplete)
- **bool isConnectionInterruptProcessingComplete** ()

### 6.247.1 Constructor & Destructor Documentation

6.247.1.1 `activemq::state::ConnectionState::ConnectionState ( const Pointer< ConnectionInfo > & info )`

6.247.1.2 `virtual activemq::state::ConnectionState::~~ConnectionState ( ) [virtual]`

### 6.247.2 Member Function Documentation

6.247.2.1 `void activemq::state::ConnectionState::addSession ( const Pointer< SessionInfo > & info ) [inline]`

6.247.2.2 `void activemq::state::ConnectionState::addTempDestination ( const Pointer< DestinationInfo > & info ) [inline]`

6.247.2.3 `void activemq::state::ConnectionState::addTransactionState ( const Pointer< TransactionId > & id ) [inline]`

6.247.2.4 `void activemq::state::ConnectionState::checkShutdown ( ) const`

6.247.2.5 `const Pointer< commands::ConnectionInfo > & activemq::state::ConnectionState::getInfo ( ) const [inline]`

6.247.2.6 `StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > & activemq::state::ConnectionState::getRecoveringPullConsumers ( ) [inline]`

6.247.2.7 `const Pointer< SessionState > & activemq::state::ConnectionState::getSessionState ( const Pointer< SessionId > & id ) const [inline]`

6.247.2.8 `std::vector< Pointer< SessionState > > & activemq::state::ConnectionState::getSessionStates ( ) const [inline]`

6.247.2.9 `const StlList< Pointer< DestinationInfo > > & activemq::state::ConnectionState::getTempDesinations ( ) const [inline]`

6.247.2.10 `const Pointer< TransactionState > & activemq::state::ConnectionState::getTransactionState ( const Pointer< TransactionId > & id ) const [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

- 6.247.2.11 `std::vector< Pointer<TransactionState> >`  
`activemq::state::ConnectionState::getTransactionStates ( ) const` `[inline]`
- 6.247.2.12 `bool activemq::state::ConnectionState::isConnectionInterruptProcessingComplete (`  
`)` `[inline]`
- 6.247.2.13 `Pointer<SessionState> activemq::state::ConnectionState::removeSession (`  
`const Pointer< SessionId > & id )` `[inline]`
- 6.247.2.14 `void activemq::state::ConnectionState::removeTempDestination ( const Pointer<`  
`ActiveMQDestination > & destination )` `[inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

- 6.247.2.15 `Pointer<TransactionState> ac-`  
`tivemq::state::ConnectionState::removeTransactionState (`  
`const Pointer< TransactionId > & id )` `[inline]`
- 6.247.2.16 `void activemq::state::ConnectionState::reset ( const Pointer< ConnectionInfo`  
`> & info )`
- 6.247.2.17 `void activemq::state::ConnectionState::setConnectionInterruptProcessingComplete (`  
`bool connectionInterruptProcessingComplete )` `[inline]`
- 6.247.2.18 `void activemq::state::ConnectionState::shutdown ( )`
- 6.247.2.19 `std::string activemq::state::ConnectionState::toString ( ) const`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/ConnectionState.h`

## 6.248 activemq::state::ConnectionStateTracker Class Reference

```
#include <src/main/activemq/state/ConnectionStateTracker.h>
```

Inheritance diagram for `activemq::state::ConnectionStateTracker`:

### Public Member Functions

- `ConnectionStateTracker ()`
- `virtual ~ConnectionStateTracker ()`
- `Pointer< Tracked > track (const Pointer< Command > &command) throw`  
`( decaf::io::IOException )`



- void **trackBack** (const **Pointer**< **Command** > &command)
- void **restore** (const **Pointer**< **transport::Transport** > &transport) throw ( decaf::io::IOException )
- void **connectionInterruptProcessingComplete** (**transport::Transport** \*transport, const **Pointer**< **ConnectionId** > &connectionId)
- void **transportInterrupted** ()
- virtual **Pointer**< **Command** > **processDestinationInfo** (**DestinationInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer**< **Command** > **processRemoveDestination** (**DestinationInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer**< **Command** > **processProducerInfo** (**ProducerInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer**< **Command** > **processRemoveProducer** (**ProducerId** \*id) throw ( exceptions::ActiveMQException )
- virtual **Pointer**< **Command** > **processConsumerInfo** (**ConsumerInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer**< **Command** > **processRemoveConsumer** (**ConsumerId** \*id) throw ( exceptions::ActiveMQException )
- virtual **Pointer**< **Command** > **processSessionInfo** (**SessionInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer**< **Command** > **processRemoveSession** (**SessionId** \*id) throw ( exceptions::ActiveMQException )
- virtual **Pointer**< **Command** > **processConnectionInfo** (**ConnectionInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer**< **Command** > **processRemoveConnection** (**ConnectionId** \*id) throw ( exceptions::ActiveMQException )
- virtual **Pointer**< **Command** > **processMessage** (**Message** \*message) throw ( exceptions::ActiveMQException )
- virtual **Pointer**< **Command** > **processMessageAck** (**MessageAck** \*ack) throw ( exceptions::ActiveMQException )
- virtual **Pointer**< **Command** > **processBeginTransaction** (**TransactionInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer**< **Command** > **processPrepareTransaction** (**TransactionInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer**< **Command** > **processCommitTransactionOnePhase** (**TransactionInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer**< **Command** > **processCommitTransactionTwoPhase** (**TransactionInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer**< **Command** > **processRollbackTransaction** (**TransactionInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer**< **Command** > **processEndTransaction** (**TransactionInfo** \*info) throw ( exceptions::ActiveMQException )
- bool **isRestoreConsumers** () const
- void **setRestoreConsumers** (bool restoreConsumers)
- bool **isRestoreProducers** () const
- void **setRestoreProducers** (bool restoreProducers)
- bool **isRestoreSessions** () const
- void **setRestoreSessions** (bool restoreSessions)

- bool **isTrackTransactions** () const
- void **setTrackTransactions** (bool trackTransactions)
- bool **isRestoreTransaction** () const
- void **setRestoreTransaction** (bool restoreTransaction)
- bool **isTrackMessages** () const
- void **setTrackMessages** (bool trackMessages)
- int **getMaxCacheSize** () const
- void **setMaxCacheSize** (int maxCacheSize)
- bool **isTrackTransactionProducers** () const
- void **setTrackTransactionProducers** (bool trackTransactionProducers)

## Friends

- class **RemoveTransactionAction**

## 6.248.1 Constructor & Destructor Documentation

6.248.1.1 `activemq::state::ConnectionStateTracker::ConnectionStateTracker ( )`

6.248.1.2 `virtual activemq::state::ConnectionStateTracker::~~ConnectionStateTracker ( )`  
[virtual]

## 6.248.2 Member Function Documentation

6.248.2.1 `void activemq::state::ConnectionStateTracker::connectionInterruptProcessingComplete ( transport::Transport * transport, const Pointer< ConnectionId > & connectionId )`

6.248.2.2 `int activemq::state::ConnectionStateTracker::getMaxCacheSize ( ) const`  
[inline]

6.248.2.3 `bool activemq::state::ConnectionStateTracker::isRestoreConsumers ( ) const`  
[inline]

6.248.2.4 `bool activemq::state::ConnectionStateTracker::isRestoreProducers ( ) const`  
[inline]

6.248.2.5 `bool activemq::state::ConnectionStateTracker::isRestoreSessions ( ) const`  
[inline]

6.248.2.6 `bool activemq::state::ConnectionStateTracker::isRestoreTransaction ( ) const`  
[inline]

6.248.2.7 `bool activemq::state::ConnectionStateTracker::isTrackMessages ( ) const`  
[inline]

6.248.2.8 `bool activemq::state::ConnectionStateTracker::isTrackTransactionProducers ( ) const`  
[inline]

6.248.2.9 `bool activemq::state::ConnectionStateTracker::isTrackTransactions ( ) const`  
[inline]

6.248.2.10 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processBeginTransaction ( TransactionInfo * info ) throw ( exceptions::ActiveMQException )`  
[virtual]

Implements `activemq::state::CommandVisitor` (p. 1235).

**6.248.2.11** `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processCommitTransactionOnePhase ( TransactionInfo * info ) throw ( exceptions::ActiveMQException )`  
[virtual]

Implements `activemq::state::CommandVisitor` (p. 1236).

**6.248.2.12** `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processCommitTransactionTwoPhase ( TransactionInfo * info ) throw ( exceptions::ActiveMQException )`  
[virtual]

Implements `activemq::state::CommandVisitor` (p. 1236).

**6.248.2.13** `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processConnectionInfo ( ConnectionInfo * info ) throw ( exceptions::ActiveMQException )`  
[virtual]

Implements `activemq::state::CommandVisitor` (p. 1236).

**6.248.2.14** `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processConsumerInfo ( ConsumerInfo * info ) throw ( exceptions::ActiveMQException )`  
[virtual]

Implements `activemq::state::CommandVisitor` (p. 1237).

**6.248.2.15** `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processDestinationInfo ( DestinationInfo * info ) throw ( exceptions::ActiveMQException )`  
[virtual]

Implements `activemq::state::CommandVisitor` (p. 1237).

**6.248.2.16** `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processEndTransaction ( TransactionInfo * info ) throw ( exceptions::ActiveMQException )`  
[virtual]

Implements `activemq::state::CommandVisitor` (p. 1237).

**6.248.2.17** virtual **Pointer<Command>** **activemq::state::ConnectionStateTracker::processMessage** (  
    **Message** \* *message* ) throw ( **exceptions::ActiveMQException** )  
    [**virtual**]

Implements **activemq::state::CommandVisitor** (p. 1238).

**6.248.2.18** virtual **Pointer<Command>** **activemq::state::ConnectionStateTracker::processMessageAck** (  
    **MessageAck** \* *ack* ) throw ( **exceptions::ActiveMQException** )  
    [**virtual**]

Implements **activemq::state::CommandVisitor** (p. 1238).

**6.248.2.19** virtual **Pointer<Command>** **activemq::state::ConnectionStateTracker::processPrepareTransaction** (  
    **TransactionInfo** \* *info* ) throw ( **exceptions::ActiveMQException** )  
    [**virtual**]

Implements **activemq::state::CommandVisitor** (p. 1239).

**6.248.2.20** virtual **Pointer<Command>** **activemq::state::ConnectionStateTracker::processProducerInfo** (  
    **ProducerInfo** \* *info* ) throw ( **exceptions::ActiveMQException** )  
    [**virtual**]

Implements **activemq::state::CommandVisitor** (p. 1239).

**6.248.2.21** virtual **Pointer<Command>** **activemq::state::ConnectionStateTracker::processRemoveConnection** (  
    **ConnectionId** \* *id* ) throw ( **exceptions::ActiveMQException** )  
    [**virtual**]

Implements **activemq::state::CommandVisitor** (p. 1239).

**6.248.2.22** virtual **Pointer<Command>** **activemq::state::ConnectionStateTracker::processRemoveConsumer** ( **ConsumerId**  
    \* *id* ) throw ( **exceptions::ActiveMQException** ) [virtual]

Implements **activemq::state::CommandVisitor** (p. 1240).

**6.248.2.23** `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveDestination ( DestinationInfo * info ) throw ( exceptions::ActiveMQException )`  
[virtual]

Implements `activemq::state::CommandVisitor` (p. 1240).

**6.248.2.24** `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveProducer ( ProducerId * id ) throw ( exceptions::ActiveMQException )`  
[virtual]

Implements `activemq::state::CommandVisitor` (p. 1240).

**6.248.2.25** `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveSession ( SessionId * id ) throw ( exceptions::ActiveMQException )` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1240).

**6.248.2.26** `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRollbackTransaction ( TransactionInfo * info ) throw ( exceptions::ActiveMQException )`  
[virtual]

Implements `activemq::state::CommandVisitor` (p. 1241).

**6.248.2.27** `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processSessionInfo ( SessionInfo * info ) throw ( exceptions::ActiveMQException )`  
[virtual]

Implements `activemq::state::CommandVisitor` (p. 1241).

- 6.248.2.28 `void activemq::state::ConnectionStateTracker::restore ( const Pointer< transport::Transport > & transport ) throw ( decaf::io::IOException )`
- 6.248.2.29 `void activemq::state::ConnectionStateTracker::setMaxCacheSize ( int maxCacheSize ) [inline]`
- 6.248.2.30 `void activemq::state::ConnectionStateTracker::setRestoreConsumers ( bool restoreConsumers ) [inline]`
- 6.248.2.31 `void activemq::state::ConnectionStateTracker::setRestoreProducers ( bool restoreProducers ) [inline]`
- 6.248.2.32 `void activemq::state::ConnectionStateTracker::setRestoreSessions ( bool restoreSessions ) [inline]`
- 6.248.2.33 `void activemq::state::ConnectionStateTracker::setRestoreTransaction ( bool restoreTransaction ) [inline]`
- 6.248.2.34 `void activemq::state::ConnectionStateTracker::setTrackMessages ( bool trackMessages ) [inline]`
- 6.248.2.35 `void activemq::state::ConnectionStateTracker::setTrackTransactionProducers ( bool trackTransactionProducers ) [inline]`
- 6.248.2.36 `void activemq::state::ConnectionStateTracker::setTrackTransactions ( bool trackTransactions ) [inline]`
- 6.248.2.37 `Pointer<Tracked> activemq::state::ConnectionStateTracker::track ( const Pointer< Command > & command ) throw ( decaf::io::IOException )`
- 6.248.2.38 `void activemq::state::ConnectionStateTracker::trackBack ( const Pointer< Command > & command )`
- 6.248.2.39 `void activemq::state::ConnectionStateTracker::transportInterrupted ( )`

### 6.248.3 Friends And Related Function Documentation

- 6.248.3.1 `friend class RemoveTransactionAction [friend]`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/ConnectionStateTracker.h`

## 6.249 decaf::util::logging::ConsoleHandler Class Reference

This **Handler** (p. 2042) publishes log records to System.err.

```
#include <src/main/decaf/util/logging/ConsoleHandler.h>
```

Inheritance diagram for `decaf::util::logging::ConsoleHandler`:

## Public Member Functions

- **ConsoleHandler** ()
- virtual **~ConsoleHandler** ()
- virtual void **close** () throw ( `decaf::io::IOException` )  
*Close the current output stream.*
- virtual void **publish** (const **LogRecord** &record)  
*Publish the Log Record to this **Handler** (p. 2042).*

### 6.249.1 Detailed Description

This **Handler** (p. 2042) publishes log records to `System.err`. By default the **SimpleFormatter** (p. 3604) is used to generate brief summaries.

Configuration: By default each **ConsoleHandler** (p. 1439) is initialized using the following **LogManager** (p. 2480) configuration properties. If properties are not defined (or have invalid values) then the specified default values are used.

`ConsoleHandler.level` specifies the default level for the **Handler** (p. 2042) (defaults to **Level.INFO** (p. 2408)). `ConsoleHandler.filter` specifies the name of a **Filter** (p. 1949) class to use (defaults to no **Filter** (p. 1949)). `ConsoleHandler.formatter` specifies the name of a **Formatter** (p. 2027) class to use (defaults to **SimpleFormatter** (p. 3604)).

Since

1.0

### 6.249.2 Constructor & Destructor Documentation

6.249.2.1 `decaf::util::logging::ConsoleHandler::ConsoleHandler ( )`

6.249.2.2 `virtual decaf::util::logging::ConsoleHandler::~~ConsoleHandler ( )` [`inline`, `virtual`]

### 6.249.3 Member Function Documentation

6.249.3.1 `virtual void decaf::util::logging::ConsoleHandler::close ( )` throw ( `decaf::io::IOException` ) [`virtual`]

Close the current output stream.

Override the **StreamHandler** (p. 3756) `close` to flush the Std Err stream but doesn't close.



### Exceptions

<i>IOException</i>	
--------------------	--

Reimplemented from **decaf::util::logging::StreamHandler** (p. 3758).

**6.249.3.2** `virtual void decaf::util::logging::ConsoleHandler::publish ( const LogRecord & record ) [virtual]`

Publish the Log Record to this **Handler** (p. 2042).

### Parameters

<i>record</i>	The <b>LogRecord</b> (p. 2487) to Publish
---------------	---

Reimplemented from **decaf::util::logging::StreamHandler** (p. 3759).

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**ConsoleHandler.h**

## 6.250 activemq::commands::ConsumerControl Class Reference

```
#include <src/main/activemq/commands/ConsumerControl.h>
```

Inheritance diagram for **activemq::commands::ConsumerControl**:

### Public Member Functions

- **ConsumerControl** ()
- virtual **~ConsumerControl** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ConsumerControl \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &**destination**)
- virtual bool **isClose** () const
- virtual void **setClose** (bool **close**)
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &**consumerId**)
- virtual int **getPrefetch** () const
- virtual void **setPrefetch** (int **prefetch**)
- virtual bool **isFlush** () const
- virtual void **setFlush** (bool **flush**)
- virtual bool **isStart** () const
- virtual void **setStart** (bool **start**)
- virtual bool **isStop** () const
- virtual void **setStop** (bool **stop**)
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** \*visitor)  
throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_CONSUMERCONTROL** = 17

### Protected Attributes

- **Pointer**< **ActiveMQDestination** > **destination**
- bool **close**
- **Pointer**< **ConsumerId** > **consumerId**
- int **prefetch**
- bool **flush**
- bool **start**
- bool **stop**

## 6.250.1 Constructor & Destructor Documentation

6.250.1.1 `activemq::commands::ConsumerControl::ConsumerControl ( )`

6.250.1.2 `virtual activemq::commands::ConsumerControl::~~ConsumerControl ( )`  
[virtual]

## 6.250.2 Member Function Documentation

6.250.2.1 `virtual ConsumerControl* activemq::commands::ConsumerControl::cloneDataStructure ( ) const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1714).

6.250.2.2 `virtual void activemq::commands::ConsumerControl::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

6.250.2.3 `virtual bool activemq::commands::ConsumerControl::equals ( const DataStructure * value ) const` [virtual]

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

6.250.2.4 **virtual const Pointer<ConsumerId>& activemq::commands::ConsumerControl::getConsumerId ( ) const**  
[virtual]

6.250.2.5 **virtual Pointer<ConsumerId>& activemq::commands::ConsumerControl::getConsumerId ( )**  
[virtual]

6.250.2.6 **virtual unsigned char activemq::commands::ConsumerControl::getDataStructureType ( ) const** [virtual]

Get the unique identifier that this object and its own Marshaler share.

### Returns

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

- 6.250.2.7 **virtual const Pointer<ActiveMQDestination>&**  
**activemq::commands::ConsumerControl::getDestination ( ) const** [virtual]
- 6.250.2.8 **virtual Pointer<ActiveMQDestination>&**  
**activemq::commands::ConsumerControl::getDestination ( )** [virtual]
- 6.250.2.9 **virtual int activemq::commands::ConsumerControl::getPrefetch ( ) const**  
[virtual]
- 6.250.2.10 **virtual bool activemq::commands::ConsumerControl::isClose ( ) const**  
[virtual]
- 6.250.2.11 **virtual bool activemq::commands::ConsumerControl::isFlush ( ) const**  
[virtual]
- 6.250.2.12 **virtual bool activemq::commands::ConsumerControl::isStart ( ) const**  
[virtual]
- 6.250.2.13 **virtual bool activemq::commands::ConsumerControl::isStop ( ) const**  
[virtual]
- 6.250.2.14 **virtual void activemq::commands::ConsumerControl::setClose ( bool *close* )**  
[virtual]
- 6.250.2.15 **virtual void activemq::commands::ConsumerControl::setConsumerId ( const**  
**Pointer< ConsumerId > & *consumerId* )** [virtual]
- 6.250.2.16 **virtual void activemq::commands::ConsumerControl::setDestination ( const**  
**Pointer< ActiveMQDestination > & *destination* )** [virtual]
- 6.250.2.17 **virtual void activemq::commands::ConsumerControl::setFlush ( bool *flush* )**  
[virtual]
- 6.250.2.18 **virtual void activemq::commands::ConsumerControl::setPrefetch ( int *prefetch* )**  
[virtual]
- 6.250.2.19 **virtual void activemq::commands::ConsumerControl::setStart ( bool *start* )**  
[virtual]
- 6.250.2.20 **virtual void activemq::commands::ConsumerControl::setStop ( bool *stop* )**  
[virtual]
- 6.250.2.21 **virtual std::string activemq::commands::ConsumerControl::toString ( ) const**  
[virtual]

Returns a string containing the information for this **DataSet** (p. 1713) such as its type and value of its elements.

**Returns**

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 770).

**6.250.2.22** `virtual Pointer<Command> activemq::commands::ConsumerControl::visit ( activemq::state::CommandVisitor * visitor ) throw ( exceptions::ActiveMQException )` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

**Returns**

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 1232).

**6.250.3 Field Documentation**

**6.250.3.1** `bool activemq::commands::ConsumerControl::close` [protected]

**6.250.3.2** `Pointer<ConsumerId> activemq::commands::ConsumerControl::consumerId` [protected]

**6.250.3.3** `Pointer<ActiveMQDestination> activemq::commands::ConsumerControl::destination` [protected]

**6.250.3.4** `bool activemq::commands::ConsumerControl::flush` [protected]

**6.250.3.5** `const unsigned char activemq::commands::ConsumerControl::ID_CONSUMERCONTROL = 17` [static]

**6.250.3.6** `int activemq::commands::ConsumerControl::prefetch` [protected]

**6.250.3.7** `bool activemq::commands::ConsumerControl::start` [protected]

**6.250.3.8** `bool activemq::commands::ConsumerControl::stop` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConsumerControl.h`

6.251

activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller

Class Reference

6.251 ~~activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller~~<sup>1447</sup>

## Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1447).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConsumerControlMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller:

## Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.251.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1447).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.251.2 Constructor & Destructor Documentation

6.251.2.1 **activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::ConsumerControlMarshaller**  
 ( ) [inline]

6.251.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::~~ConsumerControlMarshaller**  
 ( ) [inline, virtual]

### 6.251.3 Member Function Documentation

6.251.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.251.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.251.3.3 **virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::looseMarshal**  
 ( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters



## 6.251

**activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller**

**Class Reference**

**1449**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 808).

**6.251.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 809).

**6.251.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 810).

```
6.251.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 811).

```
6.251.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

6.252

**activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller**

**Class Reference**

**1451**

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 813).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ConsumerControlMarshaller.h**

## 6.252 **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1451).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller**:

### Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.252.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1451).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.252.2 Constructor & Destructor Documentation

6.252.2.1 **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::ConsumerControlMarshaller**  
( ) [inline]

6.252.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::~~ConsumerControlMarshaller**  
( ) [inline, virtual]

### 6.252.3 Member Function Documentation

6.252.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.252.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

6.252

**activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller**

**Class Reference**

**1453**

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.252.3.3** virtual void **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::looseMarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 772).

**6.252.3.4** virtual void **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 774).

```

6.252.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 775).

```

6.252.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 776).

6.253

activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller

Class Reference

1455

6.252.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

wireFormat	- describes the wire format of the broker.
dataStructure	- Object to be un-marshaled.
dataIn	- BinaryReader that provides that data.
bs	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

IOException	if an error occurs during the unmarshal.
-------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h

## 6.253 activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1455).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConsumerControlMarshaller>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller**:

#### Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.253.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1455).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.253.2 Constructor & Destructor Documentation

6.253.2.1 **activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::ConsumerControlMarshaller** ( ) [inline]

6.253.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::~~ConsumerControlMarshaller** ( ) [inline, virtual]

### 6.253.3 Member Function Documentation

6.253.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.



6.253

**activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller**

**Class Reference**

**1457**

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.253.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.253.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 780).

**6.253.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 781).

```
6.253.3.5  virtual int activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 782).

```
6.253.3.6  virtual void activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

## 6.254

**activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller**

**Class Reference**

**1459**

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 783).

**6.253.3.7** virtual void **activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ConsumerControlMarshaller.h**

## 6.254 **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1459).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller**:

## Public Member Functions

- **ConsumerControlMarshaller ()**
- virtual **~ConsumerControlMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.254.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1459).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.254

activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller

Class Reference

1461

## 6.254.2 Constructor & Destructor Documentation

6.254.2.1 `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::ConsumerControlMarshaller ( ) [inline]`

6.254.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::~~ConsumerControlMarshaller ( ) [inline, virtual]`

## 6.254.3 Member Function Documentation

6.254.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.254.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.254.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 787).

**6.254.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 788).

**6.254.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

6.254

**activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller**

**Class Reference**

**1463**

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 789).

**6.254.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 790).

**6.254.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 792).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller.h`

## 6.255 `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1464).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller`:

### Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )



6.255

**activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller**  
**Class Reference**

1465

---

*Write a object instance to data output stream.*

### 6.255.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1464).  
NOTE!: This file is auto generated - do not modify! if you need to make a change,  
please see the Java Classes in the activemq-openwire-generator module

### 6.255.2 Constructor & Destructor Documentation

6.255.2.1 **activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::ConsumerControlMarshaller**  
( ) [inline]

6.255.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::~~ConsumerControlMarshaller**  
( ) [inline, virtual]

### 6.255.3 Member Function Documentation

6.255.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.255.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.255.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure**  
\* **dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 794).

**6.255.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 795).

**6.255.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

6.255

**activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller****Class Reference****1467****Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 796).

**6.255.3.6** virtual void **activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::tightMarshal2**  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
 \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*,  
**utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** )  
 [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 797).

**6.255.3.7** virtual void **activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::tightUnmarshal**  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream**  
 \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h`

## 6.256 **activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1468).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ConsumerControlMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller**:

**Public Member Functions**

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

## 6.256

**activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller**

### Class Reference

1469

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.256.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1468).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.256.2 Constructor & Destructor Documentation

6.256.2.1 **activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::ConsumerControlMarshaller**  
( ) [inline]

6.256.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::~~ConsumerControlMarshaller**  
( ) [inline, virtual]

### 6.256.3 Member Function Documentation

6.256.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.256.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.256.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 801).

**6.256.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 802).

6.256

activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller

Class Reference

1471

```
6.256.3.5 virtual int activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 803).

```
6.256.3.6 virtual void activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 804).

```
6.256.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 806).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ConsumerControlMarshaller.h

## 6.257 activemq::commands::ConsumerId Class Reference

```
#include <src/main/activemq/commands/ConsumerId.h>
```

Inheritance diagram for **activemq::commands::ConsumerId**:

#### Public Types

- typedef **decaf::lang::PointerComparator**< **ConsumerId** > **COMPARATOR**

#### Public Member Functions

- **ConsumerId** ()
- **ConsumerId** (const **ConsumerId** &other)
- **ConsumerId** (const **SessionId** &sessionId, long long consumerId)
- virtual ~**ConsumerId** ()
- virtual unsigned char **getDataStructureType** () const

*Get the unique identifier that this object and its own Marshaller share.*



- virtual **ConsumerId** \* **cloneDataStructure** () const

*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*

- virtual void **copyDataStructure** (const **DataStructure** \*src)

*Copy the contents of the passed object into this object's members, overwriting any existing data.*

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** \*value) const

*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*

- const **Pointer**< **SessionId** > & **getParentId** () const
- virtual const std::string & **getConnectionId** () const
- virtual std::string & **getConnectionId** ()
- virtual void **setConnectionId** (const std::string &connectionId)
- virtual long long **getSessionId** () const
- virtual void **setSessionId** (long long sessionId)
- virtual long long **getValue** () const
- virtual void **setValue** (long long value)
- virtual int **compareTo** (const **ConsumerId** &value) const
- virtual bool **equals** (const **ConsumerId** &value) const
- virtual bool **operator==** (const **ConsumerId** &value) const
- virtual bool **operator<** (const **ConsumerId** &value) const
- **ConsumerId** & **operator=** (const **ConsumerId** &other)

### Static Public Attributes

- static const unsigned char **ID\_CONSUMERID** = 122

### Protected Attributes

- std::string **connectionId**
- long long **sessionId**
- long long **value**

### 6.257.1 Member Typedef Documentation

6.257.1.1 `typedef decaf::lang::PointerComparator<ConsumerId>`  
`activemq::commands::ConsumerId::COMPARATOR`

### 6.257.2 Constructor & Destructor Documentation

6.257.2.1 `activemq::commands::ConsumerId::ConsumerId ( )`

6.257.2.2 `activemq::commands::ConsumerId::ConsumerId ( const ConsumerId & other )`

6.257.2.3 `activemq::commands::ConsumerId::ConsumerId ( const SessionId & sessionId,  
long long consumerId )`

6.257.2.4 `virtual activemq::commands::ConsumerId::~~ConsumerId ( )` [virtual]

### 6.257.3 Member Function Documentation

6.257.3.1 `virtual ConsumerId* activemq::commands::ConsumerId::cloneDataStructure ( )`  
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1714).

6.257.3.2 `virtual int activemq::commands::ConsumerId::compareTo ( const ConsumerId &`  
`value ) const` [virtual]

6.257.3.3 `virtual void activemq::commands::ConsumerId::copyDataStructure ( const`  
`DataStructure * src )` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Implements `activemq::commands::DataStructure` (p. 1715).

**6.257.3.4** `virtual bool activemq::commands::ConsumerId::equals ( const DataStructure *  
value ) const` [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1716).

**6.257.3.5** `virtual bool activemq::commands::ConsumerId::equals ( const ConsumerId &  
value ) const` [virtual]

**6.257.3.6** `virtual std::string& activemq::commands::ConsumerId::getConnectionId ( )`  
[virtual]

**6.257.3.7** `virtual const std::string& activemq::commands::ConsumerId::getConnectionId ( )`  
const [virtual]

**6.257.3.8** `virtual unsigned char activemq::commands::ConsumerId::getDataStructureType ( )`  
const [virtual]

Get the unique identifier that this object and its own Marshaler share.

### Returns

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

- 6.257.3.9 `const Pointer<SessionId>& activemq::commands::ConsumerId::getParentId ( ) const`
- 6.257.3.10 `virtual long long activemq::commands::ConsumerId::getSessionId ( ) const [virtual]`
- 6.257.3.11 `virtual long long activemq::commands::ConsumerId::getValue ( ) const [virtual]`
- 6.257.3.12 `virtual bool activemq::commands::ConsumerId::operator< ( const ConsumerId & value ) const [virtual]`
- 6.257.3.13 `ConsumerId& activemq::commands::ConsumerId::operator= ( const ConsumerId & other )`
- 6.257.3.14 `virtual bool activemq::commands::ConsumerId::operator== ( const ConsumerId & value ) const [virtual]`
- 6.257.3.15 `virtual void activemq::commands::ConsumerId::setConnectionId ( const std::string & connectionId ) [virtual]`
- 6.257.3.16 `virtual void activemq::commands::ConsumerId::setSessionId ( long long sessionId ) [virtual]`
- 6.257.3.17 `virtual void activemq::commands::ConsumerId::setValue ( long long value ) [virtual]`
- 6.257.3.18 `virtual std::string activemq::commands::ConsumerId::toString ( ) const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataSet` (p. 841).

### 6.257.4 Field Documentation

- 6.257.4.1 `std::string activemq::commands::ConsumerId::connectionId [protected]`
- 6.257.4.2 `const unsigned char activemq::commands::ConsumerId::ID_-CONSUMERID = 122 [static]`

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

## 6.258 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller

---

Class Reference1477

6.257.4.3 long long activemq::commands::ConsumerId::sessionId  
[protected]

6.257.4.4 long long activemq::commands::ConsumerId::value [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/ConsumerId.h

## 6.258 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller

---

Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1477).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller:

### Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**)  
throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.258.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1477). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.258.2 Constructor & Destructor Documentation

6.258.2.1 **activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::ConsumerIdMarshaller**  
 ( ) [inline]

6.258.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::~~ConsumerIdMarshaller**  
 ( ) [inline, virtual]

### 6.258.3 Member Function Documentation

6.258.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::createObject** ( )  
 const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.258.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.258.3.3** virtual void **activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::looseMarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.258.3.4** virtual void **activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.258.3.5** virtual int **activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::tightMarshal1** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.258.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.258.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.



## 6.259 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller

### Class Reference

1481

<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ConsumerIdMarshaller.h**

## 6.259 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1481).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller**:

### Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.259.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1481). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.259.2 Constructor & Destructor Documentation

6.259.2.1 **activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::ConsumerIdMarshaller**  
( ) [inline]

6.259.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::~~ConsumerIdMarshaller**  
( ) [inline, virtual]

### 6.259.3 Member Function Documentation

6.259.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.259.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

## Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.259.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.259.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

6.259.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1690).

6.259.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1698).

## 6.260 activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller

### Class Reference

1485

6.259.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h

## 6.260 activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1485).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller:

#### Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.260.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1485). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.260.2 Constructor & Destructor Documentation

6.260.2.1 **activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::ConsumerIdMarshaller**  
( ) [inline]

6.260.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::~~ConsumerIdMarshaller**  
( ) [inline, virtual]

### 6.260.3 Member Function Documentation

6.260.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.260.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

## Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.260.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.260.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.260.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.260.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



## 6.261 activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller

### Class Reference

1489

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

6.260.3.7 virtual void **activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ConsumerIdMarshaller.h**

## 6.261 activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1489).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller**:

#### Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.261.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1489). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.261.2 Constructor & Destructor Documentation

- 6.261.2.1 **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::ConsumerIdMarshaller**  
( ) [inline]
- 6.261.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::~~ConsumerIdMarshaller**  
( ) [inline, virtual]

### 6.261.3 Member Function Documentation

- 6.261.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.261.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

## Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.261.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.261.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.261.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.261.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.262 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller

### Class Reference

1493

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

6.261.3.7 virtual void **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ConsumerIdMarshaller.h**

## 6.262 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1493).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller**:

#### Public Member Functions

- ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.262.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1493). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.262.2 Constructor & Destructor Documentation

- 6.262.2.1 **activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::ConsumerIdMarshaller**  
 ( ) [inline]
- 6.262.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::~~ConsumerIdMarshaller**  
 ( ) [inline, virtual]

### 6.262.3 Member Function Documentation

- 6.262.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::createObject** ( )  
 const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.262.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.262.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.262.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.262.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.262.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



## 6.263 activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller

### Class Reference

1497

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

6.262.3.7 virtual void **activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ConsumerIdMarshaller.h**

## 6.263 activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1497).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ConsumerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller**:

#### Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.263.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1497). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.263.2 Constructor & Destructor Documentation

- 6.263.2.1 **activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::ConsumerIdMarshaller**  
( ) [inline]
- 6.263.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::~~ConsumerIdMarshaller**  
( ) [inline, virtual]

### 6.263.3 Member Function Documentation

- 6.263.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.263.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

## Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.263.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.263.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

```
6.263.3.5  virtual int activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.263.3.6  virtual void activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.263.3.7** virtual void **activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ConsumerIdMarshaller.h**

## 6.264 activemq::commands::ConsumerInfo Class Reference

```
#include <src/main/activemq/commands/ConsumerInfo.h>
```

Inheritance diagram for **activemq::commands::ConsumerInfo**:

#### Public Member Functions

- **ConsumerInfo** ()
- virtual **~ConsumerInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ConsumerInfo** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)

*Copy the contents of the passed object into this object's members, overwriting any existing data.*

- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- **Pointer< RemoveInfo > createRemoveCommand** () const
- virtual const **Pointer< ConsumerId > & getConsumerId** () const
- virtual **Pointer< ConsumerId > & getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer< ConsumerId > &consumerId**)
- virtual bool **isBrowser** () const
- virtual void **setBrowser** (bool browser)
- virtual const **Pointer< ActiveMQDestination > & getDestination** () const
- virtual **Pointer< ActiveMQDestination > & getDestination** ()
- virtual void **setDestination** (const **Pointer< ActiveMQDestination > &destination**)
- virtual int **getPrefetchSize** () const
- virtual void **setPrefetchSize** (int prefetchSize)
- virtual int **getMaximumPendingMessageLimit** () const
- virtual void **setMaximumPendingMessageLimit** (int maximumPendingMessageLimit)
- virtual bool **isDispatchAsync** () const
- virtual void **setDispatchAsync** (bool dispatchAsync)
- virtual const std::string & **getSelector** () const
- virtual std::string & **getSelector** ()
- virtual void **setSelector** (const std::string &selector)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual bool **isNoLocal** () const
- virtual void **setNoLocal** (bool noLocal)
- virtual bool **isExclusive** () const
- virtual void **setExclusive** (bool exclusive)
- virtual bool **isRetroactive** () const
- virtual void **setRetroactive** (bool retroactive)
- virtual unsigned char **getPriority** () const
- virtual void **setPriority** (unsigned char priority)
- virtual const std::vector< **decaf::lang::Pointer< BrokerId > > & getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer< BrokerId > > & getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer< BrokerId > > &brokerPath**)

- virtual const **Pointer**< **BooleanExpression** > & **getAdditionalPredicate** () const
- virtual **Pointer**< **BooleanExpression** > & **getAdditionalPredicate** ()
- virtual void **setAdditionalPredicate** (const **Pointer**< **BooleanExpression** > &**additionalPredicate**)
- virtual bool **isNetworkSubscription** () const
- virtual void **setNetworkSubscription** (bool **networkSubscription**)
- virtual bool **isOptimizedAcknowledge** () const
- virtual void **setOptimizedAcknowledge** (bool **optimizedAcknowledge**)
- virtual bool **isNoRangeAcks** () const
- virtual void **setNoRangeAcks** (bool **noRangeAcks**)
- virtual const std::vector< **decaf::lang::Pointer**< **ConsumerId** > > & **getNetworkConsumerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **ConsumerId** > > & **getNetworkConsumerPath** ()
- virtual void **setNetworkConsumerPath** (const std::vector< **decaf::lang::Pointer**< **ConsumerId** > > &**networkConsumerPath**)
- virtual bool **isConsumerInfo** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** \*visitor) throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_CONSUMERINFO** = 5

### Protected Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- bool **browser**
- **Pointer**< **ActiveMQDestination** > **destination**
- int **prefetchSize**
- int **maximumPendingMessageLimit**
- bool **dispatchAsync**
- std::string **selector**
- std::string **subscriptionName**
- bool **noLocal**
- bool **exclusive**
- bool **retroactive**
- unsigned char **priority**
- std::vector< **decaf::lang::Pointer**< **BrokerId** > > **brokerPath**
- **Pointer**< **BooleanExpression** > **additionalPredicate**
- bool **networkSubscription**
- bool **optimizedAcknowledge**
- bool **noRangeAcks**
- std::vector< **decaf::lang::Pointer**< **ConsumerId** > > **networkConsumerPath**

## 6.264.1 Constructor & Destructor Documentation

6.264.1.1 `activemq::commands::ConsumerInfo::ConsumerInfo ( )`

6.264.1.2 `virtual activemq::commands::ConsumerInfo::~~ConsumerInfo ( ) [virtual]`

## 6.264.2 Member Function Documentation

6.264.2.1 `virtual ConsumerInfo* activemq::commands::ConsumerInfo::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1714).

6.264.2.2 `virtual void activemq::commands::ConsumerInfo::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

6.264.2.3 `Pointer<RemoveInfo> activemq::commands::ConsumerInfo::createRemoveCommand ( ) const`

6.264.2.4 `virtual bool activemq::commands::ConsumerInfo::equals ( const DataStructure * value ) const [virtual]`

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 766).



- 6.264.2.5 `virtual const Pointer<BooleanExpression>& activemq::commands::ConsumerInfo::getAdditionalPredicate ( ) const [virtual]`
- 6.264.2.6 `virtual Pointer<BooleanExpression>& activemq::commands::ConsumerInfo::getAdditionalPredicate ( ) [virtual]`
- 6.264.2.7 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConsumerInfo::getBrokerPath ( ) const [virtual]`
- 6.264.2.8 `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConsumerInfo::getBrokerPath ( ) [virtual]`
- 6.264.2.9 `virtual Pointer<ConsumerId>& activemq::commands::ConsumerInfo::getConsumerId ( ) [virtual]`
- 6.264.2.10 `virtual const Pointer<ConsumerId>& activemq::commands::ConsumerInfo::getConsumerId ( ) const [virtual]`
- 6.264.2.11 `virtual unsigned char activemq::commands::ConsumerInfo::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

### Returns

new **DataSet** (p. 1713) type copy.

Implements **activemq::commands::DataSet** (p. 1717).

- 6.264.2.12 `virtual const Pointer<ActiveMQDestination>&  
activemq::commands::ConsumerInfo::getDestination ( ) const [virtual]`
- 6.264.2.13 `virtual Pointer<ActiveMQDestination>&  
activemq::commands::ConsumerInfo::getDestination ( ) [virtual]`
- 6.264.2.14 `virtual int activemq::commands::ConsumerInfo::getMaximumPendingMessageLimit ( ) const [virtual]`
- 6.264.2.15 `virtual const std::vector< decaf::lang::Pointer<ConsumerId> >&  
activemq::commands::ConsumerInfo::getNetworkConsumerPath ( ) const [virtual]`
- 6.264.2.16 `virtual std::vector< decaf::lang::Pointer<ConsumerId> >&  
activemq::commands::ConsumerInfo::getNetworkConsumerPath ( ) [virtual]`
- 6.264.2.17 `virtual int activemq::commands::ConsumerInfo::getPrefetchSize ( ) const [virtual]`
- 6.264.2.18 `virtual unsigned char activemq::commands::ConsumerInfo::getPriority ( ) const [virtual]`
- 6.264.2.19 `virtual const std::string& activemq::commands::ConsumerInfo::getSelector ( ) const [virtual]`
- 6.264.2.20 `virtual std::string& activemq::commands::ConsumerInfo::getSelector ( ) [virtual]`
- 6.264.2.21 `virtual const std::string& activemq::commands::ConsumerInfo::getSubscriptionName ( ) const [virtual]`
- 6.264.2.22 `virtual std::string& activemq::commands::ConsumerInfo::getSubscriptionName ( ) [virtual]`
- 6.264.2.23 `virtual bool activemq::commands::ConsumerInfo::isBrowser ( ) const [virtual]`
- 6.264.2.24 `virtual bool activemq::commands::ConsumerInfo::isConsumerInfo ( ) const [inline, virtual]`

### Returns

an answer of true to the `isConsumerInfo()` (p. 1506) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 768).

- 6.264.2.25 `virtual bool activemq::commands::ConsumerInfo::isDispatchAsync ( ) const`  
[virtual]
- 6.264.2.26 `virtual bool activemq::commands::ConsumerInfo::isExclusive ( ) const`  
[virtual]
- 6.264.2.27 `virtual bool activemq::commands::ConsumerInfo::isNetworkSubscription ( ) const`  
[virtual]
- 6.264.2.28 `virtual bool activemq::commands::ConsumerInfo::isNoLocal ( ) const`  
[virtual]
- 6.264.2.29 `virtual bool activemq::commands::ConsumerInfo::isNoRangeAcks ( ) const`  
[virtual]
- 6.264.2.30 `virtual bool activemq::commands::ConsumerInfo::isOptimizedAcknowledge ( ) const` [virtual]
- 6.264.2.31 `virtual bool activemq::commands::ConsumerInfo::isRetroactive ( ) const`  
[virtual]
- 6.264.2.32 `virtual void activemq::commands::ConsumerInfo::setAdditionalPredicate ( const Pointer< BooleanExpression > & additionalPredicate )` [virtual]
- 6.264.2.33 `virtual void activemq::commands::ConsumerInfo::setBrokerPath ( const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath )`  
[virtual]
- 6.264.2.34 `virtual void activemq::commands::ConsumerInfo::setBrowser ( bool browser )`  
[virtual]
- 6.264.2.35 `virtual void activemq::commands::ConsumerInfo::setConsumerId ( const Pointer< ConsumerId > & consumerId )` [virtual]
- 6.264.2.36 `virtual void activemq::commands::ConsumerInfo::setDestination ( const Pointer< ActiveMQDestination > & destination )` [virtual]
- 6.264.2.37 `virtual void activemq::commands::ConsumerInfo::setDispatchAsync ( bool dispatchAsync )` [virtual]
- 6.264.2.38 `virtual void activemq::commands::ConsumerInfo::setExclusive ( bool exclusive )`  
[virtual]
- 6.264.2.39 `virtual void activemq::commands::ConsumerInfo::setMaximumPendingMessageLimit ( int maximumPendingMessageLimit )` [virtual]
- 6.264.2.40 `virtual void activemq::commands::ConsumerInfo::setNetworkConsumerPath ( const std::vector< decaf::lang::Pointer< ConsumerId > > & networkConsumerPath )` [virtual]
- 6.264.2.41 `virtual void activemq::commands::ConsumerInfo::setNetworkSubscription ( bool networkSubscription )` [virtual]  
Generated on Tue Mar 8 2011 17:10:05 for activemq-cpp-3.2.5 by Doxygen
- 6.264.2.42 `virtual void activemq::commands::ConsumerInfo::setNoLocal ( bool noLocal )`  
[virtual]
- 6.264.2.43 `virtual void activemq::commands::ConsumerInfo::setNoRangeAcks ( bool noRangeAcks )` [virtual]

**Returns**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

**6.264.2.51** `virtual Pointer<Command> activemq::commands::ConsumerInfo::visit  
( activemq::state::CommandVisitor * visitor ) throw (  
exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

**Returns**

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

### 6.264.3 Field Documentation

- 6.264.3.1 **Pointer<BooleanExpression> activemq::commands::ConsumerInfo::additionalPredicate**  
[protected]
- 6.264.3.2 **std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::ConsumerInfo::brokerPath** [protected]
- 6.264.3.3 **bool activemq::commands::ConsumerInfo::browser** [protected]
- 6.264.3.4 **Pointer<ConsumerId> activemq::commands::ConsumerInfo::consumerId**  
[protected]
- 6.264.3.5 **Pointer<ActiveMQDestination> activemq::commands::ConsumerInfo::destination**  
[protected]
- 6.264.3.6 **bool activemq::commands::ConsumerInfo::dispatchAsync**  
[protected]
- 6.264.3.7 **bool activemq::commands::ConsumerInfo::exclusive** [protected]
- 6.264.3.8 **const unsigned char activemq::commands::ConsumerInfo::ID\_CONSUMERINFO = 5** [static]
- 6.264.3.9 **int activemq::commands::ConsumerInfo::maximumPendingMessageLimit**  
[protected]
- 6.264.3.10 **std::vector< decaf::lang::Pointer<ConsumerId> > activemq::commands::ConsumerInfo::networkConsumerPath**  
[protected]
- 6.264.3.11 **bool activemq::commands::ConsumerInfo::networkSubscription**  
[protected]
- 6.264.3.12 **bool activemq::commands::ConsumerInfo::noLocal** [protected]
- 6.264.3.13 **bool activemq::commands::ConsumerInfo::noRangeAcks**  
[protected]
- 6.264.3.14 **bool activemq::commands::ConsumerInfo::optimizedAcknowledge**  
[protected]
- 6.264.3.15 **int activemq::commands::ConsumerInfo::prefetchSize**  
[protected]
- 6.264.3.16 **unsigned char activemq::commands::ConsumerInfo::priority**  
[protected]
- 
- 6.264.3.17 **bool activemq::commands::ConsumerInfo::retroactive**  
Generated on Tue Mar 8 2011 17:10:05 for activemq-cpp-3.2.5 by Doxygen  
[protected]
- 6.264.3.18 **std::string activemq::commands::ConsumerInfo::selector**  
[protected]
- 6.264.3.19 **std::string activemq::commands::ConsumerInfo::subscriptionName**  
[protected]

- `src/main/activemq/commands/ConsumerInfo.h`

## 6.265 `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1510).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMar
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller`:

### Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.265.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1510). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.265.2 Constructor & Destructor Documentation

**6.265.2.1** `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::ConsumerInfoMarshaller ( ) [inline]`

**6.265.2.2** `virtual activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller ( ) [inline, virtual]`

### 6.265.3 Member Function Documentation

**6.265.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.265.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.265.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 808).

**6.265.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 809).

**6.265.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 810).

**6.265.3.6** virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::tightMarshal2  
( OpenWireFormat \* *wireFormat*, commands::DataStructure  
\* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 811).

**6.265.3.7** virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::tightUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream  
\* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 813).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ConsumerInfoMarshaller.h**

## 6.266 **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1514).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMar
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller**:

**Public Member Functions**

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.266.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1514). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.266.2 Constructor & Destructor Documentation

6.266.2.1 **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::ConsumerInfoMarshaller**  
( ) [inline]

6.266.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller**  
( ) [inline, virtual]

### 6.266.3 Member Function Documentation

6.266.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.266.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

```
6.266.3.3  virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 772).

```
6.266.3.4  virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 774).

6.266.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::tightMarshal1  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**  
(p. 775).

6.266.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::tightMarshal2  
( OpenWireFormat \* *wireFormat*, commands::DataStructure  
\* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**  
(p. 776).

```
6.266.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h

## 6.267 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1518).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMar
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller**:

#### Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.267.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1518). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.267.2 Constructor & Destructor Documentation

**6.267.2.1** **activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::ConsumerInfoMarshaller**  
( ) [inline]

**6.267.2.2** **virtual activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller**  
( ) [inline, virtual]

### 6.267.3 Member Function Documentation

**6.267.3.1** **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.267.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.267.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 780).

**6.267.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---



<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 781).

**6.267.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 782).

**6.267.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 783).

```
6.267.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h

## 6.268 activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1522).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMar
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller**:

## Public Member Functions

- **ConsumerInfoMarshaller ()**
- virtual **~ConsumerInfoMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.268.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1522). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.268.2 Constructor & Destructor Documentation

6.268.2.1 `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::ConsumerInfoMarshaller ( ) [inline]`

6.268.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller ( ) [inline, virtual]`

## 6.268.3 Member Function Documentation

6.268.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.268.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.268.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 787).

**6.268.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 788).

**6.268.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 789).

```
6.268.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 790).

```
6.268.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 792).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h

## 6.269 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1527).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller:

### Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.269.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1527). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.269.2 Constructor & Destructor Documentation

6.269.2.1 **activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::ConsumerInfoMarshaller**  
( ) [inline]

6.269.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller**  
( ) [inline, virtual]

### 6.269.3 Member Function Documentation

6.269.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::createObject** ( ) **const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.269.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.269.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.



#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 794).

**6.269.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 795).

**6.269.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 796).

```
6.269.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 797).

```
6.269.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## 6.270 activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller

### Class Reference

1531

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ConsumerInfoMarshaller.h**

## 6.270 activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1531).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ConsumerInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller**:

#### Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.270.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1531). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.270.2 Constructor & Destructor Documentation

- 6.270.2.1 **activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::ConsumerInfoMarshaller**  
 ( ) [inline]
- 6.270.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller**  
 ( ) [inline, virtual]

### 6.270.3 Member Function Documentation

- 6.270.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

- 6.270.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

## 6.270 activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller

### Class Reference 1533

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

```
6.270.3.3 virtual void activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 801).

```
6.270.3.4 virtual void activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 802).

```

6.270.3.5  virtual int activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 803).

```

6.270.3.6  virtual void activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 804).

6.270.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 806).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/ConsumerInfoMarshaller.h

## 6.271 activemq::state::ConsumerState Class Reference

```
#include <src/main/activemq/state/ConsumerState.h>
```

#### Public Member Functions

- **ConsumerState** (const **Pointer**< **ConsumerInfo** > &info)
- virtual ~**ConsumerState** ()
- std::string **toString** () const
- const **Pointer**< **ConsumerInfo** > & **getInfo** () const

### 6.271.1 Constructor & Destructor Documentation

6.271.1.1 `activemq::state::ConsumerState::ConsumerState ( const Pointer< ConsumerInfo > & info )`

6.271.1.2 `virtual activemq::state::ConsumerState::~~ConsumerState ( ) [virtual]`

### 6.271.2 Member Function Documentation

6.271.2.1 `const Pointer<ConsumerInfo>& activemq::state::ConsumerState::getInfo ( ) const [inline]`

6.271.2.2 `std::string activemq::state::ConsumerState::toString ( ) const`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/ConsumerState.h`

## 6.272 `activemq::commands::ControlCommand` Class Reference

```
#include <src/main/activemq/commands/ControlCommand.h>
```

Inheritance diagram for `activemq::commands::ControlCommand`:

### Public Member Functions

- **ControlCommand** ( )
- virtual **~ControlCommand** ( )
- virtual unsigned char **getDataStructureType** ( ) const  
*Get the unique identifier that this object and its own Marshaller share.*
- virtual **ControlCommand \* cloneDataStructure** ( ) const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** ( ) const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const



*Compares the **DataSet** (p. 1713) passed in to this one, and returns if they are equivalent.*

- virtual const std::string & **getCommand** () const
- virtual std::string & **getCommand** ()
- virtual void **setCommand** (const std::string &command)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_CONTROLCOMMAND** = 14

### Protected Attributes

- std::string **command**

## 6.272.1 Constructor & Destructor Documentation

6.272.1.1 **activemq::commands::ControlCommand::ControlCommand** ( )

6.272.1.2 **virtual activemq::commands::ControlCommand::~~ControlCommand** ( )  
[virtual]

## 6.272.2 Member Function Documentation

6.272.2.1 **virtual ControlCommand\*** **activemq::commands::ControlCommand::cloneDataSet**  
( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataSet** (p. 1714).

6.272.2.2 **virtual void** **activemq::commands::ControlCommand::copyDataSet** ( const  
**DataSet** \* src ) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

**Parameters**

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

**6.272.2.3** `virtual bool activemq::commands::ControlCommand::equals ( const DataStructure * value ) const` [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

**6.272.2.4** `virtual std::string& activemq::commands::ControlCommand::getCommand ( )` [virtual]

**6.272.2.5** `virtual const std::string& activemq::commands::ControlCommand::getCommand ( ) const` [virtual]

**6.272.2.6** `virtual unsigned char activemq::commands::ControlCommand::getDataStructureType ( ) const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns**

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

**6.272.2.7** `virtual void activemq::commands::ControlCommand::setCommand ( const std::string & command )` [virtual]

**6.272.2.8** `virtual std::string activemq::commands::ControlCommand::toString ( ) const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

**Returns**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

## 6.273

**activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller**

### Class Reference

1539

6.272.2.9 **virtual** **Pointer**<**Command**> **activemq::commands::ControlCommand::visit**  
( **activemq::state::CommandVisitor** \* *visitor* ) **throw** (  
**exceptions::ActiveMQException** ) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

### Returns

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

## 6.272.3 Field Documentation

6.272.3.1 **std::string** **activemq::commands::ControlCommand::command**  
[protected]

6.272.3.2 **const unsigned char** **activemq::commands::ControlCommand::ID\_**  
**CONTROLCOMMAND = 14** [static]

The documentation for this class was generated from the following file:

- **src/main/activemq/commands/ControlCommand.h**

## 6.273 **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1539).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller**:

### Public Member Functions

- **ControlCommandMarshaller** ()
- **virtual** **~ControlCommandMarshaller** ()
- **virtual** **commands::DataStructure** \* **createObject** () **const**  
*Creates a new instance of this marshalable type.*
- **virtual** **unsigned char** **getDataStructureType** () **const**

*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.273.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1539).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.273

**activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller**

Class Reference

1541

### 6.273.2 Constructor & Destructor Documentation

6.273.2.1 **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::ControlCommandMarshaller**  
( ) [inline]

6.273.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::~~ControlCommandMarshaller**  
( ) [inline, virtual]

### 6.273.3 Member Function Documentation

6.273.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.273.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.273.3.3 **virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 808).

**6.273.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 809).

**6.273.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

6.273

**activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller**

**Class Reference**

1543

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 810).

**6.273.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 811).

**6.273.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 813).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h`

## 6.274 `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1544).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandM
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller`:

### Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )



---

*Write a object instance to data output stream.*

### 6.274.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1544).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.274.2 Constructor & Destructor Documentation

6.274.2.1 **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::ControlCommandMarshaller**  
( ) [inline]

6.274.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::~~ControlCommandMarshaller**  
( ) [inline, virtual]

### 6.274.3 Member Function Documentation

6.274.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.274.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.274.3.3 **virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 772).

**6.274.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 774).

**6.274.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

6.274

**activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller****Class Reference****1547****Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 775).

**6.274.3.6** virtual void **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::tightMarshal2**  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
 \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*,  
**utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** )  
 [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 776).

**6.274.3.7** virtual void **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::tightUnmarshal**  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream**  
 \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h`

## 6.275 **activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1548).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller**:

**Public Member Functions**

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

6.275

**activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller**

**Class Reference**

1549

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.275.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1548).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.275.2 Constructor & Destructor Documentation

6.275.2.1 **activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::ControlCommandMarshaller**  
( ) [inline]

6.275.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::~~ControlCommandMarshaller**  
( ) [inline, virtual]

### 6.275.3 Member Function Documentation

6.275.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.275.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.275.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 780).

**6.275.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 781).

6.275

activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller

Class Reference

1551

```
6.275.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 782).

```
6.275.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 783).

```
6.275.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h

## 6.276 activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1552).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller**:

#### Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*



## 6.276

**activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller**

### Class Reference

1553

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.276.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1552).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.276.2 Constructor & Destructor Documentation

6.276.2.1 **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::ControlCommandMarshaller**  
( ) [inline]

6.276.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::~~ControlCommandMarshaller**  
( ) [inline, virtual]

### 6.276.3 Member Function Documentation

6.276.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.276.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::getDataStructureType( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.276.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 787).

**6.276.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

## 6.276

**activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller**

### Class Reference

1555

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 788).

```
6.276.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 789).

```
6.276.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 790).

**6.276.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 792).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h`

## 6.277 **activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1556).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller**:

- **ControlCommandMarshaller ()**
- virtual **~ControlCommandMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.277.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1556).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.277.2 Constructor & Destructor Documentation

6.277.2.1 `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::ControlCommandMarshaller ( ) [inline]`

6.277.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::~~ControlCommandMarshaller ( ) [inline, virtual]`

## 6.277.3 Member Function Documentation

6.277.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.277.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.277.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

6.277

**activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller****Class Reference****1559****Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 794).

**6.277.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 795).

**6.277.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 796).

```
6.277.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 797).

```
6.277.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 799).

The documentation for this class was generated from the following file:



6.278

activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller

Class Reference

1561

- src/main/activemq/wireformat/openwire/marshall/v5/ControlCommandMarshaller.h

## 6.278 activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1561).

```
#include <src/main/activemq/wireformat/openwire/marshall/v6/ControlCommandMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller:

### Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.278.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1561).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.278.2 Constructor & Destructor Documentation

6.278.2.1 **activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::ControlCommandMarshaller**  
 ( ) [inline]

6.278.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::~~ControlCommandMarshaller**  
 ( ) [inline, virtual]

### 6.278.3 Member Function Documentation

6.278.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.278.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.278.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::looseMarshal**  
 ( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

6.278

**activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller**

**Class Reference**

**1563**

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 801).

**6.278.3.4** virtual void **activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 802).

**6.278.3.5** virtual int **activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::tightMarshal1** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 803).

```
6.278.3.6 virtual void activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 804).

```
6.278.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 806).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ControlCommandMarshaller.h**

**6.279 decaf::util::concurrent::CountDownLatch Class Reference**

```
#include <src/main/decaf/util/concurrent/CountDownLatch.h>
```

**Public Member Functions**

- **CountDownLatch** (int count)

*Constructor.*

- virtual **~CountDownLatch** ()
- virtual void **await** () throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::Exception )

*Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted.*

- virtual bool **await** (long long timeout) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::Exception )

*Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.*

- virtual bool **await** (long long timeout, const **TimeUnit** &unit) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::Exception )

*Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.*

- virtual void **countDown** ()

*Counts down the latch, releasing all waiting threads when the count hits zero.*

- virtual int **getCount** () const

*Gets the current count.*

## 6.279.1 Constructor & Destructor Documentation

### 6.279.1.1 `decaf::util::concurrent::CountDownLatch::CountDownLatch ( int count )`

Constructor.

#### Parameters

<i>count</i>	- number to count down from.
--------------	------------------------------

### 6.279.1.2 `virtual decaf::util::concurrent::CountDownLatch::~~CountDownLatch ( )` [virtual]

## 6.279.2 Member Function Documentation

### 6.279.2.1 `virtual void decaf::util::concurrent::CountDownLatch::await ( ) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::Exception )` [virtual]

Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted.

If the current count is zero then this method returns immediately.

If the current count is greater than zero then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happen:

- \* The count reaches zero due to invocations of the **countDown()** (p. 1568) method; or
- \* Some other thread interrupts the current thread.

If the current thread:

- \* has its interrupted status set on entry to this method; or
  - \* is interrupted while waiting,
- then `InterruptedException` is thrown and the current thread's interrupted status is cleared.

#### Exceptions

<i>InterruptedException</i>	- if the current thread is interrupted while waiting.
<i>Exception</i>	- if any other error occurs.

### 6.279.2.2 `virtual bool decaf::util::concurrent::CountDownLatch::await ( long long timeOut ) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::Exception )` [virtual]

Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.

If the current count is zero then this method returns immediately with the value `true`.

If the current count is greater than zero then the current thread becomes disabled for

thread scheduling purposes and lies dormant until one of three things happen:

\* The count reaches zero due to invocations of the **countDown()** (p. 1568) method;  
or \* Some other thread interrupts the current thread; or \* The specified waiting time elapses.

If the count reaches zero then the method returns with the value true.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while waiting,  
then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

#### Parameters

<i>timeout</i>	- Time in milliseconds to wait for the count to reach zero.
----------------	---

#### Exceptions

<i>InterruptedException</i>	- if the current thread is interrupted while waiting.
<i>Exception</i>	- if any other error occurs.

**6.279.2.3** `virtual bool decaf::util::concurrent::CountDownLatch::await ( long long timeout, const TimeUnit & unit ) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::Exception ) [virtual]`

Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.

If the current count is zero then this method returns immediately with the value true.

If the current count is greater than zero then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happen:

\* The count reaches zero due to invocations of the **countDown()** (p. 1568) method;  
or \* Some other thread interrupts the current thread; or \* The specified waiting time elapses.

If the count reaches zero then the method returns with the value true.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while waiting,  
then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

#### Parameters

<i>timeout</i>	- Time to wait for the count to reach zero.
<i>unit</i>	- The units that the timeout specifies.

**Exceptions**

<i>InterruptedException</i>	- if the current thread is interrupted while waiting.
<i>Exception</i>	- if any other error occurs.

**6.279.2.4** `virtual void decaf::util::concurrent::CountDownLatch::countDown ( )`  
`[virtual]`

Counts down the latch, releasing all waiting threads when the count hits zero.

**6.279.2.5** `virtual int decaf::util::concurrent::CountDownLatch::getCount ( ) const`  
`[inline, virtual]`

Gets the current count.

**Returns**

int count value

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/CountDownLatch.h`

**6.280 decaf::util::zip::CRC32 Class Reference**

Class that can be used to compute a CRC-32 checksum for a data stream.

```
#include <src/main/decaf/util/zip/CRC32.h>
```

Inheritance diagram for `decaf::util::zip::CRC32`:

**Public Member Functions**

- **CRC32** ()
- virtual **~CRC32** ()
- virtual long long **getValue** () const
- virtual void **reset** ()  
*Reset the checksum to its initial value.*
- virtual void **update** (const std::vector< unsigned char > &buffer)  
*Updates the current checksum with the specified vector of bytes.*
- virtual void **update** (const std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )



*Updates the current checksum with the specified array of bytes.*

- virtual void **update** (const unsigned char \*buffer, int size, int offset, int length)  
throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Updates the current checksum with the specified array of bytes.*

- virtual void **update** (int byte)

*Updates the current checksum with the specified byte value.*

### 6.280.1 Detailed Description

Class that can be used to compute a CRC-32 checksum for a data stream.

Since

1.0

### 6.280.2 Constructor & Destructor Documentation

6.280.2.1 decaf::util::zip::CRC32::CRC32 ( )

6.280.2.2 virtual decaf::util::zip::CRC32::~~CRC32 ( ) [virtual]

### 6.280.3 Member Function Documentation

6.280.3.1 virtual long long decaf::util::zip::CRC32::getValue ( ) const [virtual]

**Returns**

the current checksum value.

Implements **decaf::util::zip::Checksum** (p. 1175).

6.280.3.2 virtual void decaf::util::zip::CRC32::reset ( ) [virtual]

Reset the checksum to its initial value.

Implements **decaf::util::zip::Checksum** (p. 1175).

6.280.3.3 virtual void decaf::util::zip::CRC32::update ( const unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Updates the current checksum with the specified array of bytes.

**Parameters**

<i>buffer</i>	The buffer to read the updated bytes from.
<i>size</i>	The size of the passed buffer.
<i>offset</i>	The position in the buffer to start reading.
<i>length</i>	The amount of data to read from the byte buffer.

**Exceptions**

<i>NullPointerException</i>	if the passed buffer is NULL.
<i>IndexOutOfBoundsException</i>	if offset + length > size of the buffer.

Implements **decaf::util::zip::Checksum** (p. 1175).

**6.280.3.4** `virtual void decaf::util::zip::CRC32::update ( const std::vector< unsigned char > & buffer, int offset, int length ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Updates the current checksum with the specified array of bytes.

**Parameters**

<i>buffer</i>	The buffer to read the updated bytes from.
<i>offset</i>	The position in the buffer to start reading.
<i>length</i>	The amount of data to read from the byte buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if offset + length > size of the buffer.
----------------------------------	--

Implements **decaf::util::zip::Checksum** (p. 1176).

**6.280.3.5** `virtual void decaf::util::zip::CRC32::update ( const std::vector< unsigned char > & buffer ) [virtual]`

Updates the current checksum with the specified vector of bytes.

**Parameters**

<i>buffer</i>	The buffer to read the updated bytes from.
---------------	--

Implements **decaf::util::zip::Checksum** (p. 1176).

**6.280.3.6** `virtual void decaf::util::zip::CRC32::update ( int byte ) [virtual]`

Updates the current checksum with the specified byte value.

### Parameters

<i>byte</i>	The byte value to update the current <b>Checksum</b> (p. 1174) with (0..255).
-------------	---

Implements **decaf::util::zip::Checksum** (p. 1176).

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**CRC32.h**

## 6.281 ct\_data\_s Struct Reference

```
#include <src/main/decaf/internal/util/zip/deflate.h>
```

### Data Fields

- union {  
    **ush freq**  
    **ush code**  
} **fc**
- union {  
    **ush dad**  
    **ush len**  
} **dl**

#### 6.281.1 Field Documentation

6.281.1.1 **ush ct\_data\_s::code**

6.281.1.2 **ush ct\_data\_s::dad**

6.281.1.3 union { ... } **ct\_data\_s::dl**

6.281.1.4 union { ... } **ct\_data\_s::fc**

6.281.1.5 **ush ct\_data\_s::freq**

6.281.1.6 **ush ct\_data\_s::len**

The documentation for this struct was generated from the following file:

- src/main/decaf/internal/util/zip/**deflate.h**

## 6.282 activemq::commands::DataArrayResponse Class Reference

```
#include <src/main/activemq/commands/DataArrayResponse.h>
```

Inheritance diagram for activemq::commands::DataArrayResponse:

### Public Member Functions

- **DataArrayResponse** ()
- virtual **~DataArrayResponse** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **DataArrayResponse \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const std::vector< **decaf::lang::Pointer**< **DataStructure** > > & **getData** () const
- virtual std::vector< **decaf::lang::Pointer**< **DataStructure** > > & **getData** ()
- virtual void **setData** (const std::vector< **decaf::lang::Pointer**< **DataStructure** > > &data)

### Static Public Attributes

- static const unsigned char **ID\_DATAARRAYRESPONSE** = 33

### Protected Attributes

- std::vector< **decaf::lang::Pointer**< **DataStructure** > > **data**

## 6.282.1 Constructor & Destructor Documentation

6.282.1.1 `activemq::commands::DataArrayResponse::DataArrayResponse ( )`

6.282.1.2 `virtual activemq::commands::DataArrayResponse::~~DataArrayResponse ( )`  
[virtual]

## 6.282.2 Member Function Documentation

6.282.2.1 `virtual DataArrayResponse* activemq::commands::DataArrayResponse::cloneDataStructure ( ) const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Reimplemented from `activemq::commands::Response` (p. 3380).

6.282.2.2 `virtual void activemq::commands::DataArrayResponse::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<code>src</code>	- Source Object
------------------	-----------------

Reimplemented from `activemq::commands::Response` (p. 3380).

6.282.2.3 `virtual bool activemq::commands::DataArrayResponse::equals ( const DataStructure * value ) const` [virtual]

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::Response` (p. 3381).

- 6.282.2.4 `virtual std::vector< decaf::lang::Pointer<DataStructure> >& activemq::commands::DataArrayResponse::getData ( ) [virtual]`
- 6.282.2.5 `virtual const std::vector< decaf::lang::Pointer<DataStructure> >& activemq::commands::DataArrayResponse::getData ( ) const [virtual]`
- 6.282.2.6 `virtual unsigned char activemq::commands::DataArrayResponse::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1713) type copy.

Reimplemented from **activemq::commands::Response** (p. 3381).

- 6.282.2.7 `virtual void activemq::commands::DataArrayResponse::setData ( const std::vector< decaf::lang::Pointer< DataStructure > > & data ) [virtual]`
- 6.282.2.8 `virtual std::string activemq::commands::DataArrayResponse::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 3382).

### 6.282.3 Field Documentation

- 6.282.3.1 `std::vector< decaf::lang::Pointer<DataStructure> > activemq::commands::DataArrayResponse::data [protected]`
- 6.282.3.2 `const unsigned char activemq::commands::DataArrayResponse::ID_ - DATAARRAYRESPONSE = 33 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DataArrayResponse.h`

## 6.283 activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1574).

6.283

activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller

Class Reference

1575

#include <src/main/activemq/wireformat/openwire/marshal/v2/DataArrayResponseMarshal

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller:

## Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.283.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1574).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.283.2 Constructor & Destructor Documentation

6.283.2.1 **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::DataArrayResponseMarshaller**  
( ) [inline]

6.283.2.2 **virtual activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::~~DataArrayResponseMarshaller**  
( ) [inline, virtual]

## 6.283.3 Member Function Documentation

6.283.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3395).

6.283.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3395).

6.283.3.3 **virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



6.283

**activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller**  
**Class Reference** **1577**  
**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3395).

**6.283.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3396).

**6.283.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3397).

```
6.283.3.6 virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3397).

```
6.283.3.7 virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3398).

The documentation for this class was generated from the following file:

6.284

**activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller**

**Class Reference**

1579

- `src/main/activemq/wireformat/openwire/marshal/v2/DataArrayResponseMarshaller.h`

## 6.284 **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** **Class Reference**

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1579).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/DataArrayResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller`:

### Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.284.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1579).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.284.2 Constructor & Destructor Documentation

6.284.2.1 **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::DataArrayResponseMarshaller**  
 ( ) [inline]

6.284.2.2 **virtual activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::~~DataArrayResponseMarshaller**  
 ( ) [inline, virtual]

### 6.284.3 Member Function Documentation

6.284.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller**  
 (p. 3405).

6.284.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller**  
 (p. 3405).

6.284

activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller

Class Reference

1581

6.284.3.3 virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::looseMarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure  
\* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw (  
decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller**  
(p. 3405).

6.284.3.4 virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::looseUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure  
\* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw (  
decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller**  
(p. 3406).

6.284.3.5 virtual int activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::tightMarshal1  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3407).

```
6.284.3.6 virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3407).

```
6.284.3.7 virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

6.285

**activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller**  
**Class Reference** **1583**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3408).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**DataArrayResponseMarshaller.h**

**6.285   activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller**  
**Class Reference**

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1583).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/DataArrayResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller**:

**Public Member Functions**

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.285.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1583).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.285.2 Constructor & Destructor Documentation

6.285.2.1 **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::DataArrayResponseMarshaller**  
( ) [inline]

6.285.2.2 **virtual activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::~DataArrayResponseMarshaller**  
( ) [inline, virtual]

### 6.285.3 Member Function Documentation

6.285.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3390).



6.285

activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller

Class Reference

1585

6.285.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaller.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3390).

6.285.3.3 virtual void activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::looseMarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3390).

6.285.3.4 virtual void activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::looseUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3391).

```
6.285.3.5  virtual int activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3392).

```
6.285.3.6  virtual void activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

6.286

## activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller

### Class Reference

### Exceptions

1587

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3392).

6.285.3.7 virtual void activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3393).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**DataArrayResponseMarshaller.h**

## 6.286 activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller

### Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1587).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller**:

### Public Member Functions

- **DataArrayResponseMarshaller** ()

- virtual `~DataArrayResponseMarshaller ()`
- virtual `commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.286.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1587).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.286

activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller

Class Reference

1589

## 6.286.2 Constructor & Destructor Documentation

6.286.2.1 `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::DataArrayResponseMarshaller ( ) [inline]`

6.286.2.2 `virtual activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::~~DataArrayResponseMarshaller ( ) [inline, virtual]`

## 6.286.3 Member Function Documentation

6.286.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3410).

6.286.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3410).

6.286.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3410).

**6.286.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3411).

**6.286.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

6.286

**activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller**

**Class Reference**

**1591**

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3412).

**6.286.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3412).

**6.286.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3413).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h`

## 6.287 `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1592).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/DataArrayResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller`:

### Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual `~DataArrayResponseMarshaller` ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )



6.287

**activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller**  
**Class Reference** **1593**

---

*Write a object instance to data output stream.*

### 6.287.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1592).  
NOTE!: This file is auto generated - do not modify! if you need to make a change,  
please see the Java Classes in the activemq-openwire-generator module

### 6.287.2 Constructor & Destructor Documentation

6.287.2.1 **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::DataArrayResponseMarshaller**  
( ) [inline]

6.287.2.2 **virtual activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::~~DataArrayResponseMarshaller**  
( ) [inline, virtual]

### 6.287.3 Member Function Documentation

6.287.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller**  
(p. 3400).

6.287.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller**  
(p. 3400).

**6.287.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3400).

**6.287.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3401).

**6.287.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

6.287

**activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller****Class Reference****1595****Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3402).

```
6.287.3.6 virtual void activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3402).

```
6.287.3.7 virtual void activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3403).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**DataArrayResponseMarshaller.h**

## 6.288 activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1596).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/DataArrayResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller**:

### Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )

## 6.288

### activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller Class Reference 1597

---

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

#### 6.288.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1596).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

#### 6.288.2 Constructor & Destructor Documentation

6.288.2.1 **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::DataArrayResponseMarshaller**  
( ) [**inline**]

6.288.2.2 **virtual activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::~~DataArrayResponseMarshaller**  
( ) [**inline**, **virtual**]

#### 6.288.3 Member Function Documentation

6.288.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::createObject**  
( ) **const** [**virtual**]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3415).

**6.288.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3415).

**6.288.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3415).

**6.288.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

6.288

**activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller****Class Reference****1599****Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3416).

**6.288.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3417).

**6.288.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3417).

```
6.288.3.7  virtual void activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
  * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3418).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**DataArrayResponseMarshaller.h**

**6.289 decaf::util::zip::DataFormatException Class Reference**

```
#include <src/main/decaf/util/zip/DataFormatException.h>
```

Inheritance diagram for decaf::util::zip::DataFormatException:

**Public Member Functions**

- **DataFormatException** () throw ()  
*Default Constructor.*
- **DataFormatException** (const lang::Exception &ex) throw ()  
*Copy Constructor.*



- **DataFormatException** (const **DataFormatException** &ex) throw ()  
*Copy Constructor.*
- **DataFormatException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **DataFormatException** (const std::exception \*cause) throw ()  
*Constructor.*
- **DataFormatException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **DataFormatException** \* clone () const  
*Clones this exception.*
- virtual ~**DataFormatException** () throw ()

## 6.289.1 Constructor & Destructor Documentation

6.289.1.1 decaf::util::zip::DataFormatException::DataFormatException ( ) throw ()  
[inline]

Default Constructor.

6.289.1.2 decaf::util::zip::DataFormatException::DataFormatException ( const lang::Exception & ex ) throw () [inline]

Copy Constructor.

### Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

6.289.1.3 decaf::util::zip::DataFormatException::DataFormatException ( const DataFormatException & ex ) throw () [inline]

Copy Constructor.

### Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

**6.289.1.4** `decaf::util::zip::DataFormatException::DataFormatException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.  
Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.289.1.5** `decaf::util::zip::DataFormatException::DataFormatException ( const std::exception * cause ) throw ()` `[inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.289.1.6** `decaf::util::zip::DataFormatException::DataFormatException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()` `[inline]`

Constructor.

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.289.1.7** `virtual decaf::util::zip::DataFormatException::~~DataFormatException ( ) throw ()` `[inline, virtual]`

## 6.289.2 Member Function Documentation

**6.289.2.1** `virtual DataFormatException* decaf::util::zip::DataFormatException::clone ( )` `const` `[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

a new instance of an Exception that is a copy of this instance.

Reimplemented from **decaf::lang::Exception** (p. 1889).

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**DataFormatException.h**

## 6.290 decaf::io::DataInput Class Reference

The **DataInput** (p. 1603) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types.

```
#include <src/main/decaf/io/DataInput.h>
```

### Public Member Functions

- virtual **~DataInput** ()
- virtual bool **readBoolean** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.*
- virtual char **readByte** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads and returns one input byte.*
- virtual unsigned char **readUnsignedByte** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.*
- virtual char **readChar** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads an input char and returns the char value.*
- virtual double **readDouble** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads eight input bytes and returns a double value.*
- virtual float **readFloat** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads four input bytes and returns a float value.*

- virtual int **readInt** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads four input bytes and returns an int value.*
- virtual long long **readLong** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads eight input bytes and returns a long value.*
- virtual short **readShort** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads two input bytes and returns a short value.*
- virtual unsigned short **readUnsignedShort** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads two input bytes and returns an int value in the range 0 through 65535.*
- virtual std::string **readString** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads an NULL terminated ASCII string to the stream and returns the string to the caller.*
- virtual std::string **readLine** ()=0 throw ( decaf::io::IOException )  
*Reads the next line of text from the input stream.*
- virtual std::string **readUTF** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException, decaf::io::UTFDataFormatException )  
*Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException.*
- virtual void **readFully** (unsigned char \*buffer, int size)=0 throw ( decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads some bytes from an input stream and stores them into the buffer array buffer.*
- virtual void **readFully** (unsigned char \*buffer, int size, int offset, int length)=0 throw ( decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads length bytes from an input stream.*
- virtual long long **skipBytes** (long long num)=0 throw ( io::IOException )  
*Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes.*

### 6.290.1 Detailed Description

The **DataInput** (p. 1603) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types. There is also a facility for reconstructing Strings from data in the Java standard modified UTF-8 format.

It is generally true of all the reading routines in this interface that if end of file is reached before the desired number of bytes has been read, an **EOFException** (p. 1881) is thrown. If any byte cannot be read for any reason other than end of file, an **IOException** (p. 2210) other than **EOFException** (p. 1881) is thrown. for example, an **IOException** (p. 2210) may be thrown if the underlying input stream has been closed.

See also

**DataOutput** (p. 1622)

**DataInputStream** (p. 1612)

Since

1.0

### 6.290.2 Constructor & Destructor Documentation

6.290.2.1 virtual decaf::io::DataInput::~DataInput ( ) [inline, virtual]

### 6.290.3 Member Function Documentation

6.290.3.1 virtual bool decaf::io::DataInput::readBoolean ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [pure virtual]

Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.

Returns

the boolean value of the read in byte (0=false, 1=true).

Exceptions

<b>IOException</b> (p. 2210)	if an I/O Error occurs.
<b>EOFException</b> (p. 1881)	if the end of input is reached.

6.290.3.2 virtual char decaf::io::DataInput::readByte ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [pure virtual]

Reads and returns one input byte.

The byte is treated as a signed value in the range -128 through 127, inclusive.

**Returns**

the 8-bit value read.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.

### 6.290.3.3 `virtual char decaf::io::DataInput::readChar ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [pure virtual]`

Reads an input char and returns the char value.

A ascii char is made up of one bytes. This returns the same result as `readByte`

**Returns**

the 8 bit char read.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.

### 6.290.3.4 `virtual double decaf::io::DataInput::readDouble ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [pure virtual]`

Reads eight input bytes and returns a double value.

It does this by first constructing a long long value in exactly the manner of the `readlong` method, then converting this long value to a double in exactly the manner of the method `Double::longBitsToDouble`.

**Returns**

the double value read.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.

**6.290.3.5** virtual float decaf::io::DataInput::readFloat ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [pure virtual]

Reads four input bytes and returns a float value.

It does this by first constructing an int value in exactly the manner of the readInt method, then converting this int value to a float in exactly the manner of the method Float::intBitsToFloat.

#### Returns

the float value read.

#### Exceptions

<b>IOException</b> (p. 2210)	if an I/O Error occurs.
<b>EOFException</b> (p. 1881)	if the end of input is reached.

**6.290.3.6** virtual void decaf::io::DataInput::readFully ( unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]

Reads length bytes from an input stream.

This method blocks until one of the following conditions occurs: \* length bytes of input data are available, in which case a normal return is made. \* End of file is detected, in which case an **EOFException** (p. 1881) is thrown. \* An I/O error occurs, in which case an **IOException** (p. 2210) other than **EOFException** (p. 1881) is thrown.

If buffer is NULL, a NullPointerException is thrown. If offset+length is greater than the length of the array buffer, then an IndexOutOfBoundsException is thrown. If length is zero, then no bytes are read. Otherwise, the first byte read is stored into element buffer[offset], the next one into buffer[offset+1], and so on. The number of bytes read is, at most, equal to length.

#### Parameters

<i>buffer</i>	The byte array to insert read data into.
<i>size</i>	The size in bytes of the given byte buffer.
<i>offset</i>	The location in buffer to start writing.
<i>length</i>	The number of bytes to read from the buffer.

#### Exceptions

<b>IOException</b> (p. 2210)	if an I/O Error occurs.
<b>EOFException</b> (p. 1881)	if the end of input is reached.

<i>NullPointerException</i>	if the buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length > size, or an int param is negative.

**6.290.3.7** `virtual void decaf::io::DataInput::readFully ( unsigned char * buffer, int size ) throw ( decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::IndexOutOfBoundsException )` [pure virtual]

Reads some bytes from an input stream and stores them into the buffer array *buffer*.

The number of bytes read is equal to the length of *buffer*.

This method blocks until one of the following conditions occurs: \* *buffer*'s size bytes of input data are available, in which case a normal return is made. \* End of file is detected, in which case an **EOFException** (p. 1881) is thrown. \* An I/O error occurs, in which case an **IOException** (p. 2210) other than **EOFException** (p. 1881) is thrown.

If *buffer* size is zero, then no bytes are read. Otherwise, the first byte read is stored into element *b*[0], the next one into *b*[1], and so on. If an exception is thrown from this method, then it may be that some but not all bytes of *buffer* have been updated with data from the input stream.

#### Parameters

<i>buffer</i>	The byte array to insert read data into.
<i>size</i>	The size in bytes of the given byte buffer.

#### Exceptions

<b>IOException</b> (p. 2210)	if an I/O Error occurs.
<b>EOFException</b> (p. 1881)	if the end of input is reached.
<i>IndexOutOfBoundsException</i>	if the size value is negative.

**6.290.3.8** `virtual int decaf::io::DataInput::readInt ( ) throw ( decaf::io::IOException, decaf::io::EOFException )` [pure virtual]

Reads four input bytes and returns an int value.

Let *a* be the first byte read, *b* be the second byte, *c* be the third byte, and *d* be the fourth byte. The value returned is:

$((a \& 0xff) << 24) \mid ((b \& 0xff) << 16) \mid ((c \& 0xff) << 8) \mid (d \& 0xff)$



**Returns**

the int value read.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.

**6.290.3.9** `virtual std::string decaf::io::DataInput::readLine ( ) throw ( decaf::io::IOException ) [pure virtual]`

Reads the next line of text from the input stream.

It reads successive bytes, converting each byte to an ASCII char separately, until it encounters a line terminator or end of file; the characters read are then returned as a standard String. Note that because this method processes bytes, it does not support input of the full Unicode character set.

If end of file is encountered before even one byte can be read, then an empty string is returned. Otherwise, each byte that is read is converted to type char. If the character ' ' is encountered, it is discarded and reading ceases. If the character " " is encountered, it is discarded and, if the following byte converts to the character ' ' , then that is discarded also; reading then ceases. If end of file is encountered before either of the characters ' ' and " " is encountered, reading ceases. Once reading has ceased, a String is returned that contains all the characters read and not discarded, taken in order.

**Returns**

the next line of text read from the input stream or empty string if at EOF.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
--	-------------------------

**6.290.3.10** `virtual long long decaf::io::DataInput::readLong ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [pure virtual]`

Reads eight input bytes and returns a long value.

Let a be the first byte read, b be the second byte, c be the third byte, d be the fourth byte, e be the fifth byte, f be the sixth byte, g be the seventh byte, and h be the eighth byte. The value returned is:

```
((long)(a & 0xff) << 56) | ((long)(b & 0xff) << 48) | ((long)(c & 0xff) << 40) |
((long)(d & 0xff) << 32) | ((long)(e & 0xff) << 24) | ((long)(f & 0xff) << 16) |
((long)(g & 0xff) << 8) | ((long)(h & 0xff))
```

**Returns**

the 64 bit long long read.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.

**6.290.3.11** `virtual short decaf::io::DataInput::readShort ( ) throw ( decaf::io::IOException, decaf::io::EOFException )` [pure virtual]

Reads two input bytes and returns a short value.

Let a be the first byte read and b be the second byte. The value returned is:

```
(short)((a << 8) | (b & 0xff))
```

**Returns**

the 16 bit short value read.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.

**6.290.3.12** `virtual std::string decaf::io::DataInput::readString ( ) throw ( decaf::io::IOException, decaf::io::EOFException )` [pure virtual]

Reads an NULL terminated ASCII string to the stream and returns the string to the caller.

**Returns**

string object containing the string read.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
--	-------------------------

<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.
---	---------------------------------

**6.290.3.13** `virtual unsigned char decaf::io::DataInput::readUnsignedByte ( ) throw  
( decaf::io::IOException, decaf::io::EOFException ) [pure  
virtual]`

Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.

#### Returns

the 8 bit unsigned value read.

#### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.

**6.290.3.14** `virtual unsigned short decaf::io::DataInput::readUnsignedShort ( ) throw  
( decaf::io::IOException, decaf::io::EOFException ) [pure  
virtual]`

Reads two input bytes and returns an int value in the range 0 through 65535.

Let a be the first byte read and b be the second byte. The value returned is:

$((a \& 0xff) \ll 8) | (b \& 0xff)$

#### Returns

the 16 bit unsigned short read.

#### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.

**6.290.3.15** `virtual std::string decaf::io::DataInput::readUTF ( ) throw  
( decaf::io::IOException, decaf::io::EOFException,  
decaf::io::UTFDataFormatException ) [pure virtual]`

Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException.

This method reads String value written from a Java **DataOutputStream** (p. 1628) and assumes that the length prefix the precedes the encoded UTF-8 bytes is an unsigned short, which implies that the String will be no longer than 65535 characters.

### Returns

The decoded string read from stream.

### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.
<b><i>UTFDataFormatException</i></b> (p. 4078)	if the bytes are not valid modified UTF-8 values.

**6.290.3.16** `virtual long long decaf::io::DataInput::skipBytes ( long long num ) throw (   
io::IOException ) [pure virtual]`

Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes.

However, it may skip over some smaller number of bytes, possibly zero. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. This method never throws an **EOFException** (p. 1881). The actual number of bytes skipped is returned.

### Parameters

<i>num</i>	The number of bytes to skip over.
------------	-----------------------------------

### Returns

the total number of bytes skipped.

### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
--	-------------------------

The documentation for this class was generated from the following file:

- `src/main/decaf/io/DataInput.h`

## 6.291 decaf::io::DataInputStream Class Reference

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.

```
#include <src/main/decaf/io/DataInputStream.h>
```

Inheritance diagram for decaf::io::DataInputStream:

### Public Member Functions

- **DataInputStream** (**InputStream** \*inputStream, bool own=false)  
*Creates a **DataInputStream** (p. 1612) that uses the specified underlying **InputStream** (p. 2105).*
- virtual ~**DataInputStream** ()
- virtual bool **readBoolean** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.*
- virtual char **readByte** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads and returns one input byte.*
- virtual unsigned char **readUnsignedByte** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.*
- virtual char **readChar** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads an input char and returns the char value.*
- virtual double **readDouble** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads eight input bytes and returns a double value.*
- virtual float **readFloat** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads four input bytes and returns a float value.*
- virtual int **readInt** () throw ( decaf::io::IOException, decaf::io::EOFException )

*Reads four input bytes and returns an int value.*

- virtual long long **readLong** () throw ( decaf::io::IOException, decaf::io::EOFException )

*Reads eight input bytes and returns a long value.*

- virtual short **readShort** () throw ( decaf::io::IOException, decaf::io::EOFException )

*Reads two input bytes and returns a short value.*

- virtual unsigned short **readUnsignedShort** () throw ( decaf::io::IOException, decaf::io::EOFException )

*Reads two input bytes and returns an int value in the range 0 through 65535.*

- virtual std::string **readString** () throw ( decaf::io::IOException, decaf::io::EOFException )

*Reads an NULL terminated ASCII string to the stream and returns the string to the caller.*

- virtual std::string **readLine** () throw ( decaf::io::IOException )

*Reads the next line of text from the input stream.*

- virtual std::string **readUTF** () throw ( decaf::io::IOException, decaf::io::EOFException, decaf::io::UTFDataFormatException )

*Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException.*

- virtual void **readFully** (unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads some bytes from an input stream and stores them into the buffer array buffer.*

- virtual void **readFully** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads length bytes from an input stream.*

- virtual long long **skipBytes** (long long num) throw ( io::IOException )

*Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes.*

### 6.291.1 Detailed Description

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way. An application uses a data output stream to write data that can later be read by a data input stream.

Due to the lack of garbage collection in C++ a design decision was made to add a boolean parameter to the constructor indicating if the wrapped **InputStream** (p. 2105) is owned by this object. That way creation of the underlying stream can occur in a Java like way. Ex:

```
DataInputStream (p. 1612) os = new DataInputStream (p. 1612)( new InputStream()
(p. 2107), true )
```

### Since

1.0

## 6.291.2 Constructor & Destructor Documentation

**6.291.2.1** `decaf::io::DataInputStream::DataInputStream ( InputStream * inputStream, bool own = false )`

Creates a **DataInputStream** (p. 1612) that uses the specified underlying **InputStream** (p. 2105).

### Parameters

<i>inputStream</i>	the <b>InputStream</b> (p. 2105) instance to wrap.
<i>own</i>	indicates if this class owns the wrapped string defaults to false.

**6.291.2.2** `virtual decaf::io::DataInputStream::~~DataInputStream ( ) [virtual]`

## 6.291.3 Member Function Documentation

**6.291.3.1** `virtual bool decaf::io::DataInputStream::readBoolean ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [virtual]`

Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.

### Returns

the boolean value of the read in byte (0=false, 1=true).

### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.

**6.291.3.2** `virtual char decaf::io::DataInputStream::readByte ( ) throw ( decaf::io::IOException, decaf::io::EOFException )` [virtual]

Reads and returns one input byte.

The byte is treated as a signed value in the range -128 through 127, inclusive.

#### Returns

the 8-bit value read.

#### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.

**6.291.3.3** `virtual char decaf::io::DataInputStream::readChar ( ) throw ( decaf::io::IOException, decaf::io::EOFException )` [virtual]

Reads an input char and returns the char value.

A ascii char is made up of one bytes. This returns the same result as `readByte`

#### Returns

the 8 bit char read.

#### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.

**6.291.3.4** `virtual double decaf::io::DataInputStream::readDouble ( ) throw ( decaf::io::IOException, decaf::io::EOFException )` [virtual]

Reads eight input bytes and returns a double value.

It does this by first constructing a long long value in exactly the manner of the `readlong` method, then converting this long value to a double in exactly the manner of the method `Double::longBitsToDouble`.

#### Returns

the double value read.



**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.

**6.291.3.5** virtual float decaf::io::DataInputStream::readFloat ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [virtual]

Reads four input bytes and returns a float value.

It does this by first constructing an int value in exactly the manner of the readInt method, then converting this int value to a float in exactly the manner of the method Float::intBitsToFloat.

**Returns**

the float value read.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.

**6.291.3.6** virtual void decaf::io::DataInputStream::readFully ( unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Reads length bytes from an input stream.

This method blocks until one of the following conditions occurs: \* length bytes of input data are available, in which case a normal return is made. \* End of file is detected, in which case an **EOFException** (p. 1881) is thrown. \* An I/O error occurs, in which case an **IOException** (p. 2210) other than **EOFException** (p. 1881) is thrown.

If buffer is NULL, a NullPointerException is thrown. If offset+length is greater than the length of the array buffer, then an IndexOutOfBoundsException is thrown. If length is zero, then no bytes are read. Otherwise, the first byte read is stored into element buffer[offset], the next one into buffer[offset+1], and so on. The number of bytes read is, at most, equal to length.

**Parameters**

<i>buffer</i>	The byte array to insert read data into.
<i>size</i>	The size in bytes of the given byte buffer.
<i>offset</i>	The location in buffer to start writing.
<i>length</i>	The number of bytes to read from the buffer.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.
<b><i>NullPointerException</i></b>	if the buffer is NULL.
<b><i>IndexOutOfBoundsException</i></b>	if the offset + length > size.

**6.291.3.7** `virtual void decaf::io::DataInputStream::readFully ( unsigned char * buffer,  
int size ) throw ( decaf::io::IOException, decaf::io::EOFException,  
decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Reads some bytes from an input stream and stores them into the buffer array *buffer*.

The number of bytes read is equal to the length of *buffer*.

This method blocks until one of the following conditions occurs: \* *buffer*'s size bytes of input data are available, in which case a normal return is made. \* End of file is detected, in which case an **EOFException** (p. 1881) is thrown. \* An I/O error occurs, in which case an **IOException** (p. 2210) other than **EOFException** (p. 1881) is thrown.

If *buffer* size is zero, then no bytes are read. Otherwise, the first byte read is stored into element *b*[0], the next one into *b*[1], and so on. If an exception is thrown from this method, then it may be that some but not all bytes of *buffer* have been updated with data from the input stream.

**Parameters**

<i>buffer</i>	The byte array to insert read data into.
<i>size</i>	The size in bytes of the given byte buffer.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.
<b><i>IndexOutOfBoundsException</i></b>	if the size value is negative.

**6.291.3.8** `virtual int decaf::io::DataInputStream::readInt ( ) throw ( decaf::io::IOException,  
decaf::io::EOFException )` [virtual]

Reads four input bytes and returns an int value.

Let *a* be the first byte read, *b* be the second byte, *c* be the third byte, and *d* be the fourth byte. The value returned is:

$((a \& 0xff) << 24) | ((b \& 0xff) << 16) | ((c \& 0xff) << 8) | (d \& 0xff)$

### Returns

the int value read.

### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.

**6.291.3.9** `virtual std::string decaf::io::DataInputStream::readLine ( ) throw ( decaf::io::IOException ) [virtual]`

Reads the next line of text from the input stream.

It reads successive bytes, converting each byte to an ASCII char separately, until it encounters a line terminator or end of file; the characters read are then returned as a standard String. Note that because this method processes bytes, it does not support input of the full Unicode character set.

If end of file is encountered before even one byte can be read, then an empty string is returned. Otherwise, each byte that is read is converted to type char. If the character '

' is encountered, it is discarded and reading ceases. If the character " is encountered, it is discarded and, if the following byte converts to the character '

', then that is discarded also; reading then ceases. If end of file is encountered before either of the characters '

' and " is encountered, reading ceases. Once reading has ceased, a String is returned that contains all the characters read and not discarded, taken in order.

### Returns

the next line of text read from the input stream or empty string if at EOF.

### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
--	-------------------------

**6.291.3.10** `virtual long long decaf::io::DataInputStream::readLong ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [virtual]`

Reads eight input bytes and returns a long value.

Let a be the first byte read, b be the second byte, c be the third byte, d be the fourth byte, e be the fifth byte, f be the sixth byte, g be the seventh byte, and h be the eighth

byte. The value returned is:

```
((long)(a & 0xff) << 56) | ((long)(b & 0xff) << 48) | ((long)(c & 0xff) << 40) |
((long)(d & 0xff) << 32) | ((long)(e & 0xff) << 24) | ((long)(f & 0xff) << 16) |
((long)(g & 0xff) << 8) | ((long)(h & 0xff))
```

### Returns

the 64 bit long long read.

### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.

**6.291.3.11** `virtual short decaf::io::DataInputStream::readShort ( ) throw ( decaf::io::IOException, decaf::io::EOFException )` [virtual]

Reads two input bytes and returns a short value.

Let a be the first byte read and b be the second byte. The value returned is:

```
(short)((a << 8) | (b & 0xff))
```

### Returns

the 16 bit short value read.

### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.

**6.291.3.12** `virtual std::string decaf::io::DataInputStream::readString ( ) throw ( decaf::io::IOException, decaf::io::EOFException )` [virtual]

Reads an NULL terminated ASCII string to the stream and returns the string to the caller.

### Returns

string object containing the string read.

### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
--	-------------------------

<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.
---	---------------------------------

**6.291.3.13** virtual unsigned char decaf::io::DataInputStream::readUnsignedByte ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [virtual]

Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.

#### Returns

the 8 bit unsigned value read.

#### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.

**6.291.3.14** virtual unsigned short decaf::io::DataInputStream::readUnsignedShort ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [virtual]

Reads two input bytes and returns an int value in the range 0 through 65535.

Let a be the first byte read and b be the second byte. The value returned is:

$((a \& 0xff) \ll 8) \mid (b \& 0xff)$

#### Returns

the 16 bit unsigned short read.

#### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.

**6.291.3.15** virtual std::string decaf::io::DataInputStream::readUTF ( ) throw ( decaf::io::IOException, decaf::io::EOFException, decaf::io::UTFDataFormatException ) [virtual]

Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained

all ASCII values (0-255), if so this method will throw a `UTFFormatException`.

This method reads String value written from a Java **DataOutputStream** (p. 1628) and assumes that the length prefix the precedes the encoded UTF-8 bytes is an unsigned short, which implies that the String will be no longer than 65535 characters.

### Returns

The decoded string read from stream.

### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1881)	if the end of input is reached.
<b><i>UTFDataFormatException</i></b> (p. 4078)	if the bytes are not valid modified UTF-8 values.

**6.291.3.16** `virtual long long decaf::io::DataInputStream::skipBytes ( long long num ) throw ( io::IOException )` [virtual]

Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes.

However, it may skip over some smaller number of bytes, possibly zero. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. This method never throws an **EOFException** (p. 1881). The actual number of bytes skipped is returned.

### Parameters

<i>num</i>	The number of bytes to skip over.
------------	-----------------------------------

### Returns

the total number of bytes skipped.

### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O Error occurs.
--	-------------------------

The documentation for this class was generated from the following file:

- `src/main/decaf/io/DataInputStream.h`

## 6.292 decaf::io::DataOutput Class Reference

The **DataOutput** (p. 1622) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream.

```
#include <src/main/decaf/io/DataOutput.h>
```

### Public Member Functions

- virtual **~DataOutput** ()
- virtual void **writeBoolean** (bool value)=0 throw ( decaf::io::IOException )  
*Writes a boolean to the underlying output stream as a 1-byte value.*
- virtual void **writeByte** (unsigned char value)=0 throw ( decaf::io::IOException )  
*Writes out a byte to the underlying output stream as a 1-byte value.*
- virtual void **writeShort** (short value)=0 throw ( decaf::io::IOException )  
*Writes a short to the underlying output stream as two bytes, high byte first.*
- virtual void **writeUnsignedShort** (unsigned short value)=0 throw ( decaf::io::IOException )  
*Writes a unsigned short to the bytes message stream as a 2 byte value.*
- virtual void **writeChar** (char value)=0 throw ( decaf::io::IOException )  
*Writes out a char to the underlying output stream as a one byte value If no exception is thrown, the counter written is incremented by 1.*
- virtual void **writeInt** (int value)=0 throw ( decaf::io::IOException )  
*Writes an int to the underlying output stream as four bytes, high byte first.*
- virtual void **writeLong** (long long value)=0 throw ( decaf::io::IOException )  
*Writes an 64 bit long to the underlying output stream as eight bytes, high byte first.*
- virtual void **writeFloat** (float value)=0 throw ( decaf::io::IOException )  
*Converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the underlying output stream as a 4-byte quantity, high byte first.*
- virtual void **writeDouble** (double value)=0 throw ( decaf::io::IOException )  
*Converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the underlying output stream as an 8-byte quantity, high byte first.*
- virtual void **writeBytes** (const std::string &value)=0 throw ( decaf::io::IOException )  
*Writes out the string to the underlying output stream as a sequence of bytes.*

- virtual void **writeChars** (const std::string &value)=0 throw ( decaf::io::IOException )

*Writes a string to the underlying output stream as a sequence of characters.*

- virtual void **writeUTF** (const std::string &value)=0 throw ( decaf::io::IOException, decaf::io::UTFDataFormatException )

*Writes out the string to the underlying output stream as a modeified UTF-8 encoded sequence of bytes.*

### 6.292.1 Detailed Description

The **DataOutput** (p. 1622) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream. There is also a facility for converting Strings into the Java standard modified UTF-8 format and writing the resulting series of bytes.

If a method in this interface encounters an error while writing it will throw an **IOException** (p. 2210).

See also

**DataInput** (p. 1603)

**DataOutputStream** (p. 1628)

Since

1.0

### 6.292.2 Constructor & Destructor Documentation

6.292.2.1 virtual decaf::io::DataOutput::~DataOutput ( ) [inline, virtual]

### 6.292.3 Member Function Documentation

6.292.3.1 virtual void decaf::io::DataOutput::writeBoolean ( bool *value* ) throw ( decaf::io::IOException ) [pure virtual]

Writes a boolean to the underlying output stream as a 1-byte value.

The value true is written out as the value (byte)1; the value false is written out as the value (byte)0. If no exception is thrown, the counter written is incremented by 1.

Parameters

<i>value</i>	The boolean to write as a byte (1=true, 0=false).
--------------	---



**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error is encountered.
--	---------------------------------

**6.292.3.2** `virtual void decaf::io::DataOutput::writeByte ( unsigned char value ) throw ( decaf::io::IOException )` [pure virtual]

Writes out a byte to the underlying output stream as a 1-byte value.

If no exception is thrown, the counter written is incremented by 1.

**Parameters**

<i>value</i>	The unsigned char value to write.
--------------	-----------------------------------

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error is encountered.
--	---------------------------------

**6.292.3.3** `virtual void decaf::io::DataOutput::writeBytes ( const std::string & value ) throw ( decaf::io::IOException )` [pure virtual]

Writes out the string to the underlying output stream as a sequence of bytes.

Each character in the string is written out, in sequence, by discarding its high eight bits. If no exception is thrown, the counter written is incremented by the length of value. The value written does not include a trailing null as that is not part of the sequence of bytes, if the null is needed, then use the writeChars method.

**Parameters**

<i>value</i>	The vector of bytes to write.
--------------	-------------------------------

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error is encountered.
--	---------------------------------

**6.292.3.4** `virtual void decaf::io::DataOutput::writeChar ( char value ) throw ( decaf::io::IOException )` [pure virtual]

Writes out a char to the underlying output stream as a one byte value If no exception is thrown, the counter written is incremented by 1.

**Parameters**

<i>value</i>	The signed char value to write.
--------------	---------------------------------

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error is encountered.
--	---------------------------------

**6.292.3.5** `virtual void decaf::io::DataOutput::writeChars ( const std::string & value ) throw ( decaf::io::IOException )` [pure virtual]

Writes a string to the underlying output stream as a sequence of characters.

Each character is written to the data output stream as if by the writeChar method. If no exception is thrown, the counter written is incremented by the length of value. The trailing NULL character is written by this method.

**Parameters**

<i>value</i>	The string value to write as raw bytes.
--------------	---

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error is encountered.
--	---------------------------------

**6.292.3.6** `virtual void decaf::io::DataOutput::writeDouble ( double value ) throw ( decaf::io::IOException )` [pure virtual]

Converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the underlying output stream as an 8-byte quantity, high byte first.

If no exception is thrown, the counter written is incremented by 8.

**Parameters**

<i>value</i>	The 64bit double value to write.
--------------	----------------------------------

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error is encountered.
--	---------------------------------

**6.292.3.7** `virtual void decaf::io::DataOutput::writeFloat ( float value ) throw ( decaf::io::IOException )` [pure virtual]

Converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the underlying output stream as a 4-byte quantity, high byte first.

If no exception is thrown, the counter written is incremented by 4.

**Parameters**

<i>value</i>	The 32bit floating point value to write.
--------------	--

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error is encountered.
--	---------------------------------

**6.292.3.8** virtual void decaf::io::DataOutput::writeInt ( int *value* ) throw ( decaf::io::IOException ) [pure virtual]

Writes an int to the underlying output stream as four bytes, high byte first.

If no exception is thrown, the counter written is incremented by 4.

**Parameters**

<i>value</i>	The signed integer value to write.
--------------	------------------------------------

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error is encountered.
--	---------------------------------

**6.292.3.9** virtual void decaf::io::DataOutput::writeLong ( long long *value* ) throw ( decaf::io::IOException ) [pure virtual]

Writes an 64 bit long to the underlying output stream as eight bytes, high byte first.

If no exception is thrown, the counter written is incremented by 8.

**Parameters**

<i>value</i>	The signed 64bit long value to write.
--------------	---------------------------------------

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error is encountered.
--	---------------------------------

**6.292.3.10** virtual void decaf::io::DataOutput::writeShort ( short *value* ) throw ( decaf::io::IOException ) [pure virtual]

Writes a short to the underlying output stream as two bytes, high byte first.

If no exception is thrown, the counter written is incremented by 2.

**Parameters**

<i>value</i>	The signed short value to write.
--------------	----------------------------------

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error is encountered.
--	---------------------------------

**6.292.3.11** `virtual void decaf::io::DataOutput::writeUnsignedShort ( unsigned short value )  
throw ( decaf::io::IOException )` [pure virtual]

Writes a unsigned short to the bytes message stream as a 2 byte value.

**Parameters**

<i>value</i>	The unsigned short to write to the stream.
--------------	--

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error is encountered.
--	---------------------------------

**6.292.3.12** `virtual void decaf::io::DataOutput::writeUTF ( const std::string & value ) throw (   
decaf::io::IOException, decaf::io::UTFDataFormatException )`  
[pure virtual]

Writes out the string to the underlying output stream as a modeified UTF-8 encoded sequence of bytes.

The first two bytes written are indicate its encoded length followed by the rest of the string's characters encoded as modified UTF-8. The length represent the encoded length of the data not the actual length of the string.

**Parameters**

<i>value</i>	The string value value to write as modified UTF-8.
--------------	--

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error is encountered.
<b><i>UTFDataFormatException</i></b> (p. 4078)	if the encoded size if greater than 65535

The documentation for this class was generated from the following file:

- src/main/decaf/io/**DataOutput.h**

## 6.293 decaf::io::DataOutputStream Class Reference

A data output stream lets an application write primitive Java data types to an output stream in a portable way.

```
#include <src/main/decaf/io/DataOutputStream.h>
```

Inheritance diagram for decaf::io::DataOutputStream:

### Public Member Functions

- **DataOutputStream** (**OutputStream** \*outputStream, bool own=false)  
*Creates a new data output stream to write data to the specified underlying output stream.*
- virtual ~**DataOutputStream** ()
- virtual long long **size** () const  
*Returns the current value of the counter written, the number of bytes written to this data output stream so far.*
- virtual void **writeBoolean** (bool value) throw ( **IOException** )
- virtual void **writeByte** (unsigned char value) throw ( **IOException** )
- virtual void **writeShort** (short value) throw ( **IOException** )
- virtual void **writeUnsignedShort** (unsigned short value) throw ( **IOException** )
- virtual void **writeChar** (char value) throw ( **IOException** )
- virtual void **writeInt** (int value) throw ( **IOException** )
- virtual void **writeLong** (long long value) throw ( **IOException** )
- virtual void **writeFloat** (float value) throw ( **IOException** )

- virtual void **writeDouble** (double value) throw ( IOException )
- virtual void **writeBytes** (const std::string &value) throw ( IOException )
- virtual void **writeChars** (const std::string &value) throw ( IOException )
- virtual void **writeUTF** (const std::string &value) throw ( IOException, UTF-DataFormatException )

### Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \***buffer**, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### Protected Attributes

- long long **written**
- unsigned char **buffer** [8]

### 6.293.1 Detailed Description

A data output stream lets an application write primitive Java data types to an output stream in a portable way. An application can then use a data input stream to read the data back in.

### 6.293.2 Constructor & Destructor Documentation

**6.293.2.1** `decaf::io::DataOutputStream::DataOutputStream ( OutputStream * outputStream, bool own = false )`

Creates a new data output stream to write data to the specified underlying output stream.

#### Parameters

<i>output-Stream</i>	a stream to wrap with this one.
<i>own</i>	true if this objects owns the stream that it wraps.

6.293.2.2 virtual decaf::io::DataOutputStream::~~DataOutputStream ( ) [virtual]

### 6.293.3 Member Function Documentation

6.293.3.1 virtual void decaf::io::DataOutputStream::doWriteArrayBounded ( const unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1960).

6.293.3.2 virtual void decaf::io::DataOutputStream::doWriteByte ( unsigned char *value* ) throw ( decaf::io::IOException ) [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1960).

6.293.3.3 virtual long long decaf::io::DataOutputStream::size ( ) const [inline, virtual]

Returns the current value of the counter written, the number of bytes written to this data output stream so far.

If the counter overflows, it will be wrapped to **decaf::lang::Long::MAX\_VALUE** (p. 2510).

#### Returns

the value of the written field.

- 6.293.3.4 `virtual void decaf::io::DataOutputStream::writeBoolean ( bool value ) throw ( IOException ) [virtual]`
- 6.293.3.5 `virtual void decaf::io::DataOutputStream::writeByte ( unsigned char value ) throw ( IOException ) [virtual]`
- 6.293.3.6 `virtual void decaf::io::DataOutputStream::writeBytes ( const std::string & value ) throw ( IOException ) [virtual]`
- 6.293.3.7 `virtual void decaf::io::DataOutputStream::writeChar ( char value ) throw ( IOException ) [virtual]`
- 6.293.3.8 `virtual void decaf::io::DataOutputStream::writeChars ( const std::string & value ) throw ( IOException ) [virtual]`
- 6.293.3.9 `virtual void decaf::io::DataOutputStream::writeDouble ( double value ) throw ( IOException ) [virtual]`
- 6.293.3.10 `virtual void decaf::io::DataOutputStream::writeFloat ( float value ) throw ( IOException ) [virtual]`
- 6.293.3.11 `virtual void decaf::io::DataOutputStream::writeInt ( int value ) throw ( IOException ) [virtual]`
- 6.293.3.12 `virtual void decaf::io::DataOutputStream::writeLong ( long long value ) throw ( IOException ) [virtual]`
- 6.293.3.13 `virtual void decaf::io::DataOutputStream::writeShort ( short value ) throw ( IOException ) [virtual]`
- 6.293.3.14 `virtual void decaf::io::DataOutputStream::writeUnsignedShort ( unsigned short value ) throw ( IOException ) [virtual]`
- 6.293.3.15 `virtual void decaf::io::DataOutputStream::writeUTF ( const std::string & value ) throw ( IOException, UTFDataFormatException ) [virtual]`

#### 6.293.4 Field Documentation

- 6.293.4.1 `unsigned char decaf::io::DataOutputStream::buffer[8] [protected]`
- 6.293.4.2 `long long decaf::io::DataOutputStream::written [protected]`

The documentation for this class was generated from the following file:

- `src/main/decaf/io/DataOutputStream.h`



## 6.294 activemq::commands::DataResponse Class Reference

```
#include <src/main/activemq/commands/DataResponse.h>
```

Inheritance diagram for activemq::commands::DataResponse:

### Public Member Functions

- **DataResponse** ()
- virtual **~DataResponse** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **DataResponse** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **DataStructure** > & **getData** () const
- virtual **Pointer**< **DataStructure** > & **getData** ()
- virtual void **setData** (const **Pointer**< **DataStructure** > &data)

### Static Public Attributes

- static const unsigned char **ID\_DATARESPONSE** = 32

### Protected Attributes

- **Pointer**< **DataStructure** > data

## 6.294.1 Constructor & Destructor Documentation

6.294.1.1 `activemq::commands::DataResponse::DataResponse ( )`

6.294.1.2 `virtual activemq::commands::DataResponse::~~DataResponse ( ) [virtual]`

## 6.294.2 Member Function Documentation

6.294.2.1 `virtual DataResponse* activemq::commands::DataResponse::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Reimplemented from `activemq::commands::Response` (p. 3380).

6.294.2.2 `virtual void activemq::commands::DataResponse::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from `activemq::commands::Response` (p. 3380).

6.294.2.3 `virtual bool activemq::commands::DataResponse::equals ( const DataStructure * value ) const [virtual]`

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::Response` (p. 3381).

**6.294.2.4** virtual **Pointer<DataStructure>&** **activemq::commands::DataResponse::getData** ( ) [virtual]

**6.294.2.5** virtual const **Pointer<DataStructure>&** **activemq::commands::DataResponse::getData** ( ) const [virtual]

**6.294.2.6** virtual unsigned char **activemq::commands::DataResponse::getDataStructureType** ( ) const [virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1713) type copy.

Reimplemented from **activemq::commands::Response** (p. 3381).

**6.294.2.7** virtual void **activemq::commands::DataResponse::setData** ( const **Pointer<DataStructure>** & *data* ) [virtual]

**6.294.2.8** virtual std::string **activemq::commands::DataResponse::toString** ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 3382).

### 6.294.3 Field Documentation

**6.294.3.1** **Pointer<DataStructure>** **activemq::commands::DataResponse::data** [protected]

**6.294.3.2** const unsigned char **activemq::commands::DataResponse::ID\_ - DATARESPONSE = 32** [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**DataResponse.h**

## 6.295 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller

### Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1635).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMar
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller:

### Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.295.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1635). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.295.2 Constructor & Destructor Documentation

6.295.2.1 **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::DataResponseMarshaller**  
( ) [inline]

6.295.2.2 **virtual activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::~~DataResponseMarshaller**  
( ) [inline, virtual]

### 6.295.3 Member Function Documentation

6.295.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::createObject** (  
) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3400).

6.295.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3400).

6.295.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3400).

**6.295.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3401).

**6.295.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3402).

**6.295.3.6** virtual void activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::tightMarshal2 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3402).

**6.295.3.7** virtual void activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3403).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h`

## 6.296 **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller** Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1639).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/DataResponseMar
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller**:

**Public Member Functions**

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*



- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.296.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1639). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.296.2 Constructor & Destructor Documentation

6.296.2.1 **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::DataResponseMarshaller**  
( ) [inline]

6.296.2.2 **virtual activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::~~DataResponseMarshaller**  
( ) [inline, virtual]

### 6.296.3 Member Function Documentation

6.296.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3415).

6.296.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3415).

**6.296.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3415).

**6.296.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3416).

## 6.296 activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller

### Class Reference 1643

6.296.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3417).

6.296.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3417).

```
6.296.3.7 virtual void activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
  * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3418).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**DataResponseMarshaller.h**

## 6.297 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1643).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMar
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller**:

#### Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

## 6.297.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1643). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.297.2 Constructor & Destructor Documentation

**6.297.2.1** **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::DataResponseMarshaller**  
( ) [inline]

**6.297.2.2** **virtual activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::~~DataResponseMarshaller**  
( ) [inline, virtual]

## 6.297.3 Member Function Documentation

**6.297.3.1** **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3395).

**6.297.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3395).

**6.297.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3395).

**6.297.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3396).

**6.297.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3397).

**6.297.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3397).

**6.297.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3398).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMarshaller.h`

## 6.298 **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1647).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMar
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller**:



## Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.298.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1647). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.298.2 Constructor & Destructor Documentation

6.298.2.1 `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::DataResponseMarshaller ( ) [inline]`

6.298.2.2 `virtual activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::~~DataResponseMarshaller ( ) [inline, virtual]`

## 6.298.3 Member Function Documentation

6.298.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3405).

6.298.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3405).

6.298.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3405).

**6.298.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3406).

**6.298.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3407).

```
6.298.3.6 virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3407).

```
6.298.3.7 virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3408).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMarshaller.h

## 6.299 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1652).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/DataResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller:

### Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.299.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1652). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.299.2 Constructor & Destructor Documentation

6.299.2.1 `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::DataResponseMarshaller ( ) [inline]`

6.299.2.2 `virtual activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::~DataResponseMarshaller ( ) [inline, virtual]`

### 6.299.3 Member Function Documentation

6.299.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3390).

6.299.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3390).

6.299.3.3 virtual void activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3390).

6.299.3.4 virtual void activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3391).

6.299.3.5 virtual int activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::tightMarshal1 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3392).

```
6.299.3.6 virtual void activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3392).

```
6.299.3.7 virtual void activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**



## 6.300 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller Class Reference 1657

---

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3393).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**DataResponseMarshaller.h**

## 6.300 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1656).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller**:

### Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.300.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1656). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.300.2 Constructor & Destructor Documentation

6.300.2.1 **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::DataResponseMarshaller**  
( ) [inline]

6.300.2.2 **virtual activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::~~DataResponseMarshaller**  
( ) [inline, virtual]

### 6.300.3 Member Function Documentation

6.300.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::createObject** ( ) **const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3410).

**6.300.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3410).

**6.300.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3410).

**6.300.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3411).

**6.300.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3412).

**6.300.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## 6.301 activemq::wireformat::openwire::marshal::DataStreamMarshaller Class Reference 1661

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3412).

**6.300.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3413).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h`

## 6.301 activemq::wireformat::openwire::marshal::DataStreamMarshaller Class Reference

Base class for all classes that marshal commands for Openwire.

```
#include <src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::DataStreamMarshaller**:

### Public Member Functions

- `virtual ~DataStreamMarshaller ()`

- virtual unsigned char **getDataStructureType** () const =0  
*Gets the DataStructureType that this class marshals/unmarshals.*
- virtual **commands::DataStructure \* createObject** () const =0  
*Creates a new instance of the class that this class is a marshaling director for.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*format, **commands::DataStructure** \*command, **utils::BooleanStream** \*bs)=0 throw ( **decaf::io::IOException** )  
*Tight Marshal to the given stream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*format, **commands::DataStructure** \*command, **decaf::io::DataOutputStream** \*ds, **utils::BooleanStream** \*bs)=0 throw ( **decaf::io::IOException** )  
*Tight Marshal to the given stream.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*format, **commands::DataStructure** \*command, **decaf::io::DataInputStream** \*dis, **utils::BooleanStream** \*bs)=0 throw ( **decaf::io::IOException** )  
*Tight Un-marhsal to the given stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*format, **commands::DataStructure** \*command, **decaf::io::DataOutputStream** \*ds)=0 throw ( **decaf::io::IOException** )  
*Tight Marshal to the given stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*format, **commands::DataStructure** \*command, **decaf::io::DataInputStream** \*dis)=0 throw ( **decaf::io::IOException** )  
*Loose Un-marhsal to the given stream.*

### 6.301.1 Detailed Description

Base class for all classes that marshal commands for Openwire.

### 6.301.2 Constructor & Destructor Documentation

- 6.301.2.1 virtual **activemq::wireformat::openwire::marshal::DataStreamMarshaller::~DataStreamMarshaller** ( ) [inline, virtual]

### 6.301.3 Member Function Documentation

- 6.301.3.1 virtual **commands::DataStructure\* activemq::wireformat::openwire::marshal::DataStreamMarshaller::createObject** ( ) const [pure virtual]

Creates a new instance of the class that this class is a marshaling director for.

## Returns

newly allocated Command

Implemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 195), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 240), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 370), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 398), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 446), `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 492), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 558), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 616), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 651), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 681), `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 711), `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller` (p. 887), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 920), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1316), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1350), `activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller` (p. 1382), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1414), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1461), `activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller` (p. 1490), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1524), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1553), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1589), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1657), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1798), `activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller` (p. 1832), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1920), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 2020), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2181), `activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller` (p. 2249), `activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller` (p. 2278), `activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller` (p. 2301), `activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller` (p. 2332), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2361), `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 2396), `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller` (p. 2447), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2667), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2709), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller` (p. 2739), `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller` (p. 2776), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2847), `activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller` (p. 2902), `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 3029), `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 3151), `activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller` (p. 3182), `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 3200), `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3302),

`activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller` (p. 3319), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller` (p. 3352), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3410), `activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller` (p. 3502), `activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 3518), `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3586), `activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller` (p. 3792), `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3969), `activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller` (p. 4122), `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 4161), `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 203), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 257), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 383), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 411), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 459), `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 505), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 571), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 629), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 659), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 694), `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 724), `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller` (p. 899), `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 933), `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1329), `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1337), `activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller` (p. 1370), `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1401), `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1448), `activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller` (p. 1478), `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1511), `activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1541), `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1576), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1644), `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1786), `activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller` (p. 1820), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1903), `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 2007), `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2168), `activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller` (p. 2233), `activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller` (p. 2262), `activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller` (p. 2285), `activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller` (p. 2316), `activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 2344), `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller` (p. 2383), `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller` (p. 2430), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller` (p. 2654), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller` (p. 2692), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller` (p. 2726), `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller` (p. 2756), `activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2830),



activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2882),  
activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 3011),  
activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 3130),  
activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 3162),  
activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 3196),  
activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 3289),  
activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller  
(p. 3328), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller  
(p. 3357), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 3395),  
activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 3482), ac-  
tivismq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 3527), ac-  
tivismq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 3581),  
activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 3808),  
activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3986),  
activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 4114),  
activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 4152),  
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller  
(p. 191), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller  
(p. 236), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller  
(p. 366), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller  
(p. 394), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller  
(p. 442), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller  
(p. 488), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller  
(p. 554), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller  
(p. 612), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller  
(p. 642), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller  
(p. 673), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller  
(p. 702), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 879),  
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 912), ac-  
tivismq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1308),  
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1341),  
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1374),  
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1405),  
activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1452),  
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1482),  
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1515),  
activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1545),  
activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1580),  
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1649),  
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1790),  
activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1824),  
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1907),  
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 2011),  
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 2172),  
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 2241),  
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 2266),  
activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 2289),  
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 2320),  
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 2348),  
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller

(p. 2379), `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller` (p. 2434), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller` (p. 2658), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller` (p. 2696), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller` (p. 2730), `activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller` (p. 2768), `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2838), `activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller` (p. 2894), `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 3020), `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 3138), `activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller` (p. 3170), `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 3208), `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3297), `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller` (p. 3323), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller` (p. 3361), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3405), `activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller` (p. 3498), `activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 3522), `activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3594), `activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller` (p. 3788), `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3974), `activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller` (p. 4126), `activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 4165), `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 199), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 244), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 374), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 402), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 450), `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` (p. 496), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 563), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 621), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 646), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 677), `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller` (p. 707), `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` (p. 883), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 916), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1312), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1346), `activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` (p. 1378), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1409), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1456), `activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` (p. 1486), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1519), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1549), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1584), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1653), `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1794), `activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1828), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1916), `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 2016), `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2177),

activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller (p. 2245),  
activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller (p. 2274),  
activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (p. 2297),  
activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (p. 2328),  
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 2352),  
activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller  
(p. 2392), activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller  
(p. 2443), activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller  
(p. 2662), activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller  
(p. 2704), activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller  
(p. 2734), activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (p. 2760),  
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2842),  
activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (p. 2898),  
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 3025),  
activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 3134),  
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 3166),  
activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 3191),  
activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 3310),  
activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller  
(p. 3340), activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller  
(p. 3348), activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 3390),  
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 3486), ac-  
tivismq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 3531), ac-  
tivismq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3598),  
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3800),  
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3982),  
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 4118),  
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 4157),  
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller  
(p. 208), activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller  
(p. 249), activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller  
(p. 378), activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller  
(p. 407), activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller  
(p. 454), activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller  
(p. 501), activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller  
(p. 567), activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller  
(p. 625), activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller  
(p. 655), activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller  
(p. 685), activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller  
(p. 715), activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (p. 891),  
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (p. 924), ac-  
tivismq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (p. 1321),  
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1354),  
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (p. 1386),  
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1418),  
activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1465),  
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (p. 1494),  
activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1528),  
activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1558),  
activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1593),

**activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller** (p. 1636),  
**activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller** (p. 1807),  
**activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller** (p. 1836),  
**activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1912),  
**activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller** (p. 2024),  
**activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller** (p. 2185),  
**activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller** (p. 2237),  
**activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller** (p. 2258),  
**activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller** (p. 2305),  
**activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller** (p. 2324),  
**activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller** (p. 2356),  
**activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller**  
(p. 2388), **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller**  
(p. 2439), **activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller**  
(p. 2671), **activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller**  
(p. 2700), **activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller**  
(p. 2743), **activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller** (p. 2764),  
**activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller** (p. 2834),  
**activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller** (p. 2890),  
**activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 3016),  
**activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller** (p. 3142),  
**activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller** (p. 3174),  
**activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller** (p. 3204),  
**activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller** (p. 3306),  
**activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller**  
(p. 3336), **activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller**  
(p. 3369), **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3400),  
**activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller** (p. 3494), **ac-**  
**tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller** (p. 3514), **ac-**  
**tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller** (p. 3590),  
**activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller** (p. 3796),  
**activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller** (p. 3965),  
**activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller** (p. 4106),  
**activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller** (p. 4169),  
**activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller**  
(p. 212), **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller**  
(p. 253), **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller**  
(p. 387), **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller**  
(p. 415), **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller**  
(p. 463), **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller**  
(p. 509), **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller**  
(p. 575), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller**  
(p. 633), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller**  
(p. 663), **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller**  
(p. 690), **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller**  
(p. 719), **activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller** (p. 895),  
**activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller** (p. 929), **ac-**  
**tivemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller** (p. 1325),  
**activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller** (p. 1358),  
**activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller** (p. 1390),

activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller (p. 1422),  
activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller (p. 1469),  
activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller (p. 1498),  
activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller (p. 1532),  
activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller (p. 1562),  
activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller (p. 1597),  
activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller (p. 1640),  
activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller (p. 1803),  
activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller (p. 1816),  
activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller (p. 1899),  
activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller (p. 2003),  
activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller (p. 2164),  
activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller (p. 2229),  
activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller (p. 2270),  
activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller (p. 2293),  
activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller (p. 2312),  
activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller (p. 2339),  
activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller  
(p. 2375), activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller  
(p. 2426), activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller  
(p. 2650), activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller  
(p. 2713), activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller  
(p. 2722), activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller (p. 2772),  
activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller (p. 2851),  
activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller (p. 2886),  
activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller (p. 3007),  
activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller (p. 3147),  
activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller (p. 3178),  
activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller (p. 3213),  
activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller (p. 3293),  
activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller  
(p. 3332), activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller  
(p. 3365), activemq::wireformat::openwire::marshal::v6::ResponseMarshaller (p. 3415),  
activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller (p. 3490), ac-  
tivemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller (p. 3510), ac-  
tivemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller (p. 3577),  
activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller (p. 3804),  
activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller (p. 3978),  
activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller (p. 4110),  
and activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller (p. 4148).

6.301.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::DataStreamMarshaller::getDataStructureType  
( ) const [pure virtual]

Gets the DataStructureType that this class marshals/unmarshals.

### Returns

byte Id of this classes DataStructureType

Implemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 195), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 240), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 370), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 398), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 446), `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 492), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 559), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 617), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 651), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 681), `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 711), `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller` (p. 888), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 920), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1317), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1350), `activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller` (p. 1382), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1414), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1461), `activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller` (p. 1491), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1524), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1554), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1589), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1658), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1799), `activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller` (p. 1833), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1921), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 2020), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2181), `activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller` (p. 2249), `activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller` (p. 2278), `activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller` (p. 2301), `activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller` (p. 2333), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2361), `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 2397), `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller` (p. 2447), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2667), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2709), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller` (p. 2739), `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller` (p. 2777), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2847), `activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller` (p. 2902), `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 3029), `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 3151), `activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller` (p. 3183), `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 3200), `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3302), `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller` (p. 3320), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller` (p. 3353), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3410), `activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller` (p. 3502), `ac-`

tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 3518), activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 3586), activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 3792), activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3970), activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 4122), activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 4161), activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller (p. 204), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller (p. 257), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller (p. 383), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller (p. 411), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller (p. 459), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller (p. 505), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller (p. 571), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller (p. 629), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller (p. 659), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller (p. 694), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller (p. 724), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 900), activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 933), activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1329), activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1337), activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1370), activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1401), activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1448), activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1478), activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1511), activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1541), activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1576), activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1645), activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1786), activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1821), activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1903), activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 2007), activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 2168), activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 2233), activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 2262), activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 2285), activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 2317), activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 2344), activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller (p. 2384), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller (p. 2430), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 2654), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (p. 2692), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller (p. 2726), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2757), activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2830), activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2882), activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 3011), activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 3130), activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 3163),

`activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 3196),  
`activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 3289),  
`activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller`  
(p. 3328), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller`  
(p. 3357), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3395),  
`activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller` (p. 3482), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 3527), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 3582),  
`activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller` (p. 3808),  
`activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3987),  
`activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller` (p. 4114),  
`activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 4153),  
`activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller`  
(p. 191), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller`  
(p. 236), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller`  
(p. 366), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller`  
(p. 394), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller`  
(p. 442), `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller`  
(p. 488), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller`  
(p. 554), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller`  
(p. 613), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller`  
(p. 643), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller`  
(p. 673), `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller`  
(p. 703), `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller` (p. 879),  
`activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 912), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1308),  
`activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1342),  
`activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller` (p. 1374),  
`activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1405),  
`activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1452),  
`activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller` (p. 1482),  
`activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1515),  
`activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1545),  
`activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1580),  
`activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1649),  
`activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1790),  
`activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller` (p. 1825),  
`activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1908),  
`activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 2012),  
`activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2173),  
`activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller` (p. 2241),  
`activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller` (p. 2266),  
`activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller` (p. 2289),  
`activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller` (p. 2321),  
`activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 2348),  
`activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller`  
(p. 2379), `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller`  
(p. 2435), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller`  
(p. 2658), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller`  
(p. 2696), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller`



(p. 2730), **activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller** (p. 2769),  
**activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller** (p. 2838),  
**activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller** (p. 2894),  
**activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 3020),  
**activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller** (p. 3138),  
**activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller** (p. 3171),  
**activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller** (p. 3209),  
**activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller** (p. 3298),  
**activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller**  
(p. 3324), **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller**  
(p. 3361), **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3405),  
**activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller** (p. 3498), **ac-**  
**tivemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller** (p. 3523), **ac-**  
**tivemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller** (p. 3594),  
**activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller** (p. 3788),  
**activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller** (p. 3974),  
**activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller** (p. 4126),  
**activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 4165),  
**activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller**  
(p. 199), **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller**  
(p. 245), **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller**  
(p. 374), **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller**  
(p. 403), **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller**  
(p. 450), **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller**  
(p. 497), **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller**  
(p. 563), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller**  
(p. 621), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller**  
(p. 647), **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller**  
(p. 677), **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller**  
(p. 707), **activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller** (p. 883),  
**activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller** (p. 916), **ac-**  
**tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller** (p. 1312),  
**activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller** (p. 1346),  
**activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller** (p. 1378),  
**activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller** (p. 1410),  
**activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller** (p. 1457),  
**activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller** (p. 1487),  
**activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller** (p. 1520),  
**activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller** (p. 1549),  
**activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1585),  
**activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1653),  
**activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller** (p. 1794),  
**activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller** (p. 1829),  
**activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1916),  
**activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller** (p. 2016),  
**activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 2177),  
**activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller** (p. 2245),  
**activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller** (p. 2274),  
**activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller** (p. 2297),  
**activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller** (p. 2329),

`activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2352),  
`activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller`  
(p. 2392), `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller`  
(p. 2443), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller`  
(p. 2663), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller`  
(p. 2705), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`  
(p. 2735), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2761),  
`activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2843),  
`activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2898),  
`activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 3025),  
`activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 3134),  
`activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller` (p. 3167),  
`activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 3192),  
`activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 3310),  
`activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller`  
(p. 3341), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller`  
(p. 3348), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3390),  
`activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller` (p. 3486), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 3531), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 3599),  
`activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller` (p. 3800),  
`activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3982),  
`activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller` (p. 4118),  
`activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 4157),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller`  
(p. 208), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`  
(p. 249), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller`  
(p. 379), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller`  
(p. 407), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller`  
(p. 455), `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller`  
(p. 501), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller`  
(p. 567), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller`  
(p. 625), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller`  
(p. 655), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`  
(p. 686), `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller`  
(p. 715), `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` (p. 892),  
`activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 925), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1321),  
`activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1354),  
`activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1386),  
`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1418),  
`activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1465),  
`activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1495),  
`activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1528),  
`activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1558),  
`activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1593),  
`activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1636),  
`activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1807),  
`activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1837),  
`activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1912),

activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 2024),  
activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 2186),  
activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (p. 2237),  
activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (p. 2258),  
activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (p. 2305),  
activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (p. 2325),  
activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 2357),  
activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller  
(p. 2388), activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller  
(p. 2439), activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller  
(p. 2671), activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller  
(p. 2700), activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller  
(p. 2743), activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (p. 2765),  
activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2834),  
activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (p. 2890),  
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (p. 3016),  
activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 3143),  
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (p. 3175),  
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 3204),  
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 3306),  
activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller  
(p. 3336), activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller  
(p. 3370), activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 3400),  
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller (p. 3494), ac-  
tivismq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 3514), ac-  
tivismq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 3590),  
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller (p. 3796),  
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3966),  
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller (p. 4106),  
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller (p. 4170),  
activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller  
(p. 212), activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller  
(p. 253), activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller  
(p. 387), activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller  
(p. 415), activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller  
(p. 463), activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller  
(p. 509), activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller  
(p. 576), activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller  
(p. 634), activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller  
(p. 664), activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller  
(p. 690), activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller  
(p. 720), activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller (p. 896),  
activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller (p. 929), ac-  
tivismq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller (p. 1325),  
activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller (p. 1359),  
activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller (p. 1390),  
activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller (p. 1422),  
activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller (p. 1469),  
activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller (p. 1499),  
activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller (p. 1532),

activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller (p. 1562),  
 activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller (p. 1598),  
 activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller (p. 1640),  
 activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller (p. 1803),  
 activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller (p. 1816),  
 activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller (p. 1899),  
 activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller (p. 2003),  
 activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller (p. 2164),  
 activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller (p. 2229),  
 activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller (p. 2270),  
 activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller (p. 2293),  
 activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller (p. 2313),  
 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller (p. 2340),  
 activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller  
 (p. 2375), activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller  
 (p. 2426), activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller  
 (p. 2650), activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller  
 (p. 2713), activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller  
 (p. 2722), activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller (p. 2773),  
 activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller (p. 2851),  
 activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller (p. 2886),  
 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller (p. 3007),  
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller (p. 3147),  
 activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller (p. 3179),  
 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller (p. 3213),  
 activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller (p. 3293),  
 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller  
 (p. 3332), activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller  
 (p. 3365), activemq::wireformat::openwire::marshal::v6::ResponseMarshaller (p. 3415),  
 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller (p. 3490), ac-  
 tivemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller (p. 3510), ac-  
 tivemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller (p. 3577),  
 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller (p. 3804),  
 activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller (p. 3978),  
 activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller (p. 4110),  
 and activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller (p. 4149).

```

6.301.3.3  virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::looseMarshal
( OpenWireFormat * format, commands::DataStructure * command,
  decaf::io::DataOutputStream * ds ) throw ( decaf::io::IOException )
[pure virtual]
  
```

Tight Marshal to the given stream.

#### Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

## Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 195), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 241), **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 329), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 370), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 399), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 446), **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 493), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 559), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 588), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 617), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 651), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 682), **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 711), **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 787), **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller** (p. 888), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 921), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 1317), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 1350), **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller** (p. 1382), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1414), **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller** (p. 1461), **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller** (p. 1491), **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller** (p. 1524), **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller** (p. 1554), **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1589), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1658), **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller** (p. 1799), **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller** (p. 1833), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1921), **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller** (p. 2020), **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 2182), **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller** (p. 2249), **activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller** (p. 2278), **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller** (p. 2301), **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller** (p. 2333), **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller** (p. 2361), **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 2397), **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 2448), **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller** (p. 2667), **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller** (p. 2709), **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller** (p. 2739), **activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller** (p. 2777), **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2799), **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller** (p. 2847), **activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller** (p. 2902), **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 3030),

`activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 3151),  
`activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller` (p. 3183),  
`activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 3200),  
`activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3302),  
`activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller`  
(p. 3320), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller`  
(p. 3353), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3410),  
`activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller` (p. 3502), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 3519), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3586),  
`activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller` (p. 3792),  
`activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3941),  
`activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3970),  
`activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller` (p. 4122),  
`activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 4161),  
`activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller`  
(p. 204), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller`  
(p. 258), `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller`  
(p. 341), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller`  
(p. 383), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller`  
(p. 411), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller`  
(p. 459), `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller`  
(p. 505), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller`  
(p. 572), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller`  
(p. 600), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller`  
(p. 630), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller`  
(p. 660), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller`  
(p. 694), `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller`  
(p. 724), `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller`  
(p. 808), `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller` (p. 900),  
`activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 933), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1330),  
`activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1338),  
`activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller` (p. 1370),  
`activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1401),  
`activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1448),  
`activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller` (p. 1479),  
`activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1511),  
`activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1541),  
`activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1576),  
`activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1645),  
`activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1786),  
`activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller` (p. 1821),  
`activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1904),  
`activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 2008),  
`activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2169),  
`activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller` (p. 2233),  
`activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller` (p. 2262),  
`activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller` (p. 2285),  
`activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller` (p. 2317),

activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 2344),  
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller  
(p. 2384), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller  
(p. 2431), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller  
(p. 2654), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller  
(p. 2692), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller  
(p. 2726), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2757),  
activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2790), ac-  
tivemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2830), ac-  
tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2882),  
activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 3012),  
activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 3130),  
activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 3163),  
activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 3196),  
activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 3289),  
activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller  
(p. 3328), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller  
(p. 3357), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 3395),  
activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 3482), ac-  
tivemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 3527), ac-  
tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 3582),  
activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 3808),  
activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 3945),  
activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3987),  
activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 4114),  
activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 4153),  
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller  
(p. 191), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller  
(p. 236), activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller  
(p. 325), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller  
(p. 366), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller  
(p. 394), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller  
(p. 442), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller  
(p. 488), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller  
(p. 555), activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller  
(p. 584), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller  
(p. 613), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller  
(p. 643), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller  
(p. 673), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller  
(p. 703), activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller  
(p. 772), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 880),  
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 912), ac-  
tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1308),  
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1342),  
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1374),  
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1406),  
activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1453),  
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1483),  
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1516),  
activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1545),

`activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1581),  
`activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1649),  
`activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1790),  
`activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller` (p. 1825),  
`activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1908),  
`activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 2012),  
`activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2173),  
`activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller` (p. 2241),  
`activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller` (p. 2266),  
`activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller` (p. 2289),  
`activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller` (p. 2321),  
`activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 2348),  
`activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller`  
(p. 2380), `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller`  
(p. 2435), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller`  
(p. 2659), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller`  
(p. 2696), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller`  
(p. 2731), `activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller` (p. 2769),  
`activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2786), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2839), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller` (p. 2894),  
`activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 3021),  
`activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 3139),  
`activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller` (p. 3171),  
`activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 3209),  
`activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3298),  
`activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller`  
(p. 3324), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller`  
(p. 3361), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3405),  
`activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller` (p. 3498), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 3523), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3595),  
`activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller` (p. 3788),  
`activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3949),  
`activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3974),  
`activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller` (p. 4126),  
`activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 4166),  
`activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller`  
(p. 200), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller`  
(p. 245), `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller`  
(p. 333), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller`  
(p. 375), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller`  
(p. 403), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller`  
(p. 451), `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller`  
(p. 497), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller`  
(p. 563), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller`  
(p. 592), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller`  
(p. 621), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller`  
(p. 647), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`  
(p. 677), `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller`



(p. 707), `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 780), `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` (p. 884), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 916), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1313), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1346), `activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` (p. 1378), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1410), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1457), `activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` (p. 1487), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1520), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1550), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1585), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1654), `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1795), `activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1829), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1917), `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 2016), `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2177), `activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 2245), `activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 2274), `activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` (p. 2297), `activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller` (p. 2329), `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2353), `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller` (p. 2393), `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller` (p. 2443), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2663), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2705), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller` (p. 2735), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2761), `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2795), `activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2843), `activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2898), `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 3025), `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 3134), `activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller` (p. 3167), `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 3192), `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 3311), `activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller` (p. 3341), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller` (p. 3349), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3390), `activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller` (p. 3486), `activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 3531), `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 3599), `activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller` (p. 3800), `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3953), `activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3983), `activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller` (p. 4118), `activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 4157), `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 208), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`

(p. 249), `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller`  
(p. 337), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller`  
(p. 379), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller`  
(p. 407), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller`  
(p. 455), `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller`  
(p. 501), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller`  
(p. 567), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller`  
(p. 596), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller`  
(p. 625), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller`  
(p. 655), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`  
(p. 686), `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller`  
(p. 716), `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller`  
(p. 794), `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` (p. 892),  
`activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 925), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1321),  
`activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1355),  
`activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1386),  
`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1418),  
`activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1465),  
`activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1495),  
`activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1528),  
`activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1558),  
`activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1594),  
`activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1636),  
`activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1807),  
`activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1837),  
`activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1912),  
`activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 2025),  
`activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2186),  
`activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` (p. 2237),  
`activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller` (p. 2258),  
`activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` (p. 2305),  
`activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 2325),  
`activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 2357),  
`activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller`  
(p. 2388), `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller`  
(p. 2439), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller`  
(p. 2671), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller`  
(p. 2701), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller`  
(p. 2743), `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` (p. 2765),  
`activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2781), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2834), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2890),  
`activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 3016),  
`activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 3143),  
`activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 3175),  
`activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 3205),  
`activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3306),  
`activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller`  
(p. 3337), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller`

(p. 3370), **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3400),  
**activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller** (p. 3494), **ac-**  
**tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller** (p. 3514), **ac-**  
**tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller** (p. 3590),  
**activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller** (p. 3796),  
**activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3937),  
**activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller** (p. 3966),  
**activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller** (p. 4106),  
**activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller** (p. 4170),  
**activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller**  
(p. 212), **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller**  
(p. 253), **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller**  
(p. 345), **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller**  
(p. 387), **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller**  
(p. 416), **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller**  
(p. 463), **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller**  
(p. 510), **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller**  
(p. 576), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller**  
(p. 604), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller**  
(p. 634), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller**  
(p. 664), **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller**  
(p. 690), **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller**  
(p. 720), **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller**  
(p. 801), **activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller** (p. 896),  
**activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller** (p. 929), **ac-**  
**tivemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller** (p. 1325),  
**activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller** (p. 1359),  
**activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller** (p. 1390),  
**activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller** (p. 1423),  
**activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller** (p. 1470),  
**activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller** (p. 1499),  
**activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller** (p. 1533),  
**activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller** (p. 1562),  
**activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller** (p. 1598),  
**activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller** (p. 1641),  
**activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller** (p. 1803),  
**activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller** (p. 1817),  
**activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller** (p. 1899),  
**activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller** (p. 2003),  
**activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller** (p. 2164),  
**activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller** (p. 2229),  
**activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller** (p. 2270),  
**activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller** (p. 2293),  
**activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller** (p. 2313),  
**activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller** (p. 2340),  
**activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller**  
(p. 2375), **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller**  
(p. 2426), **activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller**  
(p. 2650), **activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller**  
(p. 2713), **activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller**

(p. 2722), `activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller` (p. 2773), `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2804), `activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2851), `activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller` (p. 2886), `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 3007), `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 3147), `activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller` (p. 3179), `activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 3213), `activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 3294), `activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller` (p. 3332), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller` (p. 3366), `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3415), `activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller` (p. 3490), `activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller` (p. 3510), `activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller` (p. 3578), `activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller` (p. 3804), `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3957), `activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3978), `activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller` (p. 4110), and `activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller` (p. 4149).

```
6.301.3.4  virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::looseUnmarshal
( OpenWireFormat * format, commands::DataStructure * command,
  decaf::io::DataInputStream * dis ) throw ( decaf::io::IOException )
    [pure virtual]
```

Loose Un-marhsal to the given stream.

#### Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 196), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 241), `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 329), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 371), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 399), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 447), `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 493), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 559), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 588), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 617), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 652), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller`

(p. 682), `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller`  
(p. 712), `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller`  
(p. 788), `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller` (p. 888),  
`activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 921), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1317),  
`activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1351),  
`activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller` (p. 1383),  
`activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1415),  
`activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1462),  
`activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller` (p. 1491),  
`activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1525),  
`activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1554),  
`activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1590),  
`activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1658),  
`activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1799),  
`activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller` (p. 1833),  
`activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1921),  
`activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 2021),  
`activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2182),  
`activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller` (p. 2250),  
`activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller` (p. 2279),  
`activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller` (p. 2302),  
`activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller` (p. 2333),  
`activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2362),  
`activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller`  
(p. 2397), `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller`  
(p. 2448), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller`  
(p. 2668), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller`  
(p. 2710), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller`  
(p. 2740), `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller` (p. 2777),  
`activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2800), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2848), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller` (p. 2903),  
`activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 3030),  
`activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 3152),  
`activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller` (p. 3183),  
`activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 3201),  
`activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3303),  
`activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller`  
(p. 3320), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller`  
(p. 3353), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3411),  
`activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller` (p. 3503), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 3519), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3587),  
`activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller` (p. 3793),  
`activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3941),  
`activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3970),  
`activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller` (p. 4123),  
`activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 4162),  
`activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller`

(p. 204), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller`  
(p. 258), `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller`  
(p. 341), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller`  
(p. 384), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller`  
(p. 412), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller`  
(p. 460), `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller`  
(p. 506), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller`  
(p. 572), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller`  
(p. 600), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller`  
(p. 630), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller`  
(p. 660), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller`  
(p. 695), `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller`  
(p. 725), `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller`  
(p. 809), `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller` (p. 900),  
`activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 934), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1330),  
`activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1338),  
`activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller` (p. 1371),  
`activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1402),  
`activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1449),  
`activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller` (p. 1479),  
`activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1512),  
`activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1542),  
`activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1577),  
`activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1645),  
`activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1787),  
`activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller` (p. 1821),  
`activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1904),  
`activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 2008),  
`activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2169),  
`activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller` (p. 2234),  
`activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller` (p. 2263),  
`activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller` (p. 2286),  
`activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller` (p. 2317),  
`activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 2345),  
`activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller`  
(p. 2384), `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller`  
(p. 2431), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller`  
(p. 2655), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller`  
(p. 2693), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller`  
(p. 2727), `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller` (p. 2757),  
`activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2791), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2831), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller` (p. 2883),  
`activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 3012),  
`activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 3131),  
`activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller` (p. 3163),  
`activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 3197),  
`activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 3290),  
`activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller`

(p. 3329), **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller**  
(p. 3358), **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3396),  
**activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller** (p. 3483), **ac-**  
**tivemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller** (p. 3528), **ac-**  
**tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller** (p. 3582),  
**activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller** (p. 3809),  
**activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3945),  
**activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller** (p. 3987),  
**activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller** (p. 4115),  
**activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 4153),  
**activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller**  
(p. 192), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller**  
(p. 237), **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller**  
(p. 325), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller**  
(p. 367), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller**  
(p. 395), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller**  
(p. 443), **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller**  
(p. 489), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller**  
(p. 555), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller**  
(p. 584), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller**  
(p. 613), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller**  
(p. 643), **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller**  
(p. 674), **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller**  
(p. 703), **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**  
(p. 774), **activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller** (p. 880),  
**activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller** (p. 913), **ac-**  
**tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller** (p. 1309),  
**activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller** (p. 1342),  
**activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller** (p. 1375),  
**activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller** (p. 1406),  
**activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller** (p. 1453),  
**activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller** (p. 1483),  
**activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller** (p. 1516),  
**activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller** (p. 1546),  
**activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1581),  
**activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1650),  
**activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller** (p. 1791),  
**activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller** (p. 1825),  
**activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1908),  
**activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller** (p. 2012),  
**activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 2173),  
**activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller** (p. 2242),  
**activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller** (p. 2267),  
**activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller** (p. 2290),  
**activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller** (p. 2321),  
**activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller** (p. 2349),  
**activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller**  
(p. 2380), **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller**  
(p. 2435), **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller**  
(p. 2659), **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller**

(p. 2697), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller` (p. 2731), `activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller` (p. 2769), `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2786), `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2839), `activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller` (p. 2895), `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 3021), `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 3139), `activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller` (p. 3171), `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 3209), `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3298), `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller` (p. 3324), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller` (p. 3362), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3406), `activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller` (p. 3499), `activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 3523), `activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3595), `activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller` (p. 3789), `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3949), `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3975), `activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller` (p. 4127), `activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 4166), `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 200), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 245), `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 333), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 375), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 403), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 451), `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` (p. 497), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 564), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 592), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 622), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 647), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 678), `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller` (p. 708), `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 781), `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` (p. 884), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 917), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1313), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1347), `activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` (p. 1379), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1410), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1457), `activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` (p. 1487), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1520), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1550), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1585), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1654), `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1795), `activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1829), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1917),



activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 2017),  
activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 2178),  
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller (p. 2246),  
activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller (p. 2275),  
activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (p. 2298),  
activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (p. 2329),  
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 2353),  
activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller  
(p. 2393), activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller  
(p. 2444), activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller  
(p. 2663), activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller  
(p. 2705), activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller  
(p. 2735), activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (p. 2761),  
activemq::wireformat::openwire::marshal::v4::MessageMarshaller (p. 2795), ac-  
tivismq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2843), ac-  
tivismq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (p. 2899),  
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 3026),  
activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 3135),  
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 3167),  
activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 3192),  
activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 3311),  
activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller  
(p. 3341), activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller  
(p. 3349), activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 3391),  
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 3487), ac-  
tivismq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 3532), ac-  
tivismq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3599),  
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3801),  
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller (p. 3953),  
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3983),  
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 4119),  
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 4158),  
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller  
(p. 209), activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller  
(p. 250), activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller  
(p. 337), activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller  
(p. 379), activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller  
(p. 408), activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller  
(p. 455), activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller  
(p. 502), activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller  
(p. 568), activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller  
(p. 596), activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller  
(p. 626), activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller  
(p. 656), activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller  
(p. 686), activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller  
(p. 716), activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller  
(p. 795), activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (p. 892),  
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (p. 925), ac-  
tivismq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (p. 1322),  
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1355),

`activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1387),  
`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1419),  
`activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1466),  
`activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1495),  
`activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1529),  
`activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1559),  
`activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1594),  
`activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1637),  
`activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1808),  
`activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1837),  
`activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1913),  
`activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 2025),  
`activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2186),  
`activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` (p. 2238),  
`activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller` (p. 2259),  
`activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` (p. 2306),  
`activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 2325),  
`activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 2357),  
`activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller`  
(p. 2389), `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller`  
(p. 2440), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller`  
(p. 2672), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller`  
(p. 2701), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller`  
(p. 2744), `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` (p. 2765),  
`activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2782), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2835), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2891),  
`activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 3017),  
`activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 3143),  
`activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 3175),  
`activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 3205),  
`activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3307),  
`activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller`  
(p. 3337), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller`  
(p. 3370), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3401),  
`activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 3495), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 3515), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 3591),  
`activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller` (p. 3797),  
`activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3937),  
`activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3966),  
`activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller` (p. 4107),  
`activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 4170),  
`activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller`  
(p. 213), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller`  
(p. 254), `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller`  
(p. 345), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller`  
(p. 388), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller`  
(p. 416), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller`  
(p. 464), `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller`

(p. 510), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller`  
(p. 576), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller`  
(p. 604), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller`  
(p. 634), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller`  
(p. 664), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller`  
(p. 691), `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller`  
(p. 720), `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller`  
(p. 802), `activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller` (p. 896),  
`activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 930), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1326),  
`activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1359),  
`activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller` (p. 1391),  
`activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1423),  
`activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1470),  
`activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller` (p. 1499),  
`activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1533),  
`activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1563),  
`activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1598),  
`activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1641),  
`activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1804),  
`activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller` (p. 1817),  
`activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1900),  
`activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 2004),  
`activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2165),  
`activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller` (p. 2230),  
`activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller` (p. 2271),  
`activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller` (p. 2294),  
`activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller` (p. 2313),  
`activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2340),  
`activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller`  
(p. 2376), `activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller`  
(p. 2427), `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller`  
(p. 2651), `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller`  
(p. 2714), `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller`  
(p. 2723), `activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller` (p. 2773),  
`activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2804), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2852), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller` (p. 2887),  
`activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 3008),  
`activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 3148),  
`activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller` (p. 3179),  
`activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 3214),  
`activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 3294),  
`activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller`  
(p. 3333), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller`  
(p. 3366), `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3416),  
`activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller` (p. 3491), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller` (p. 3511), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller` (p. 3578),  
`activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller` (p. 3805),

`activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3957),  
`activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3979),  
`activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller` (p. 4111),  
and `activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller` (p. 4149).

**6.301.3.5** `virtual int activemq::wireformat::openwire::marshal::DataStreamMarshaller::tightMarshal1`  
( `OpenWireFormat * format`, `commands::DataStructure * command`,  
`utils::BooleanStream * bs` ) throw ( `decaf::io::IOException` ) [pure  
virtual]

Tight Marshal to the given stream.

#### Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>bs</i>	- boolean stream to marshal to.

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 196), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 242), `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 330), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 371), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 400), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 447), `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 494), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 560), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 589), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 618), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 652), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 683), `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 712), `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 789), `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller` (p. 889), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 922), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1318), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1351), `activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller` (p. 1383), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1415), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1462), `activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller` (p. 1492), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1525), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1555), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1590), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1659), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1800), `activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller` (p. 1834),

activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1922),  
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 2021),  
activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 2182),  
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 2250),  
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 2279),  
activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 2302),  
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 2334),  
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 2362),  
activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller  
(p. 2398), activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller  
(p. 2449), activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller  
(p. 2668), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller  
(p. 2710), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller  
(p. 2740), activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2778),  
activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2800), ac-  
tivismq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2848), ac-  
tivismq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2903),  
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 3031),  
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 3152),  
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 3184),  
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 3201),  
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 3303),  
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller  
(p. 3321), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller  
(p. 3354), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 3412),  
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 3503), ac-  
tivismq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 3520), ac-  
tivismq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 3587),  
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 3793),  
activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3942),  
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3971),  
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 4123),  
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 4162),  
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller  
(p. 205), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller  
(p. 259), activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller  
(p. 342), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller  
(p. 384), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller  
(p. 412), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller  
(p. 460), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller  
(p. 506), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller  
(p. 573), activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller  
(p. 601), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller  
(p. 631), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller  
(p. 661), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller  
(p. 695), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller  
(p. 725), activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller  
(p. 810), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 901),  
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 934), ac-  
tivismq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1331),

`activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1339),  
`activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller` (p. 1371),  
`activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1402),  
`activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1449),  
`activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller` (p. 1479),  
`activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1512),  
`activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1542),  
`activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1577),  
`activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1646),  
`activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1787),  
`activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller` (p. 1822),  
`activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1905),  
`activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 2009),  
`activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2169),  
`activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller` (p. 2234),  
`activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller` (p. 2263),  
`activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller` (p. 2286),  
`activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller` (p. 2318),  
`activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 2345),  
`activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller`  
(p. 2385), `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller`  
(p. 2432), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller`  
(p. 2655), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller`  
(p. 2693), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller`  
(p. 2727), `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller` (p. 2758),  
`activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2791), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2831), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller` (p. 2883),  
`activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 3013),  
`activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 3131),  
`activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller` (p. 3164),  
`activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 3197),  
`activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 3290),  
`activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller`  
(p. 3329), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller`  
(p. 3358), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3397),  
`activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller` (p. 3483), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 3528), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 3583),  
`activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller` (p. 3809),  
`activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3946),  
`activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3988),  
`activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller` (p. 4115),  
`activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 4154),  
`activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller`  
(p. 192), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller`  
(p. 237), `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller`  
(p. 326), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller`  
(p. 367), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller`  
(p. 395), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller`

(p. 443), `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller`  
(p. 489), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller`  
(p. 556), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller`  
(p. 585), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller`  
(p. 614), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller`  
(p. 644), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller`  
(p. 674), `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller`  
(p. 704), `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller`  
(p. 775), `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller` (p. 880),  
`activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 913), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1309),  
`activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1343),  
`activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller` (p. 1375),  
`activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1407),  
`activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1454),  
`activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller` (p. 1484),  
`activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1517),  
`activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1546),  
`activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1581),  
`activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1650),  
`activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1791),  
`activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller` (p. 1826),  
`activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1909),  
`activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 2013),  
`activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2174),  
`activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller` (p. 2242),  
`activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller` (p. 2267),  
`activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller` (p. 2290),  
`activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller` (p. 2322),  
`activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 2349),  
`activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller`  
(p. 2381), `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller`  
(p. 2436), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller`  
(p. 2660), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller`  
(p. 2697), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller`  
(p. 2732), `activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller` (p. 2770),  
`activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2787), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2840), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller` (p. 2895),  
`activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 3022),  
`activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 3140),  
`activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller` (p. 3172),  
`activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 3210),  
`activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3299),  
`activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller`  
(p. 3325), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller`  
(p. 3362), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3407),  
`activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller` (p. 3499), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 3524), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3596),

`activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller` (p. 3789),  
`activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3950),  
`activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3975),  
`activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller` (p. 4127),  
`activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 4167),  
`activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller`  
(p. 201), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller`  
(p. 246), `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller`  
(p. 334), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller`  
(p. 376), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller`  
(p. 404), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller`  
(p. 452), `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller`  
(p. 498), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller`  
(p. 564), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller`  
(p. 593), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller`  
(p. 622), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller`  
(p. 648), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`  
(p. 678), `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller`  
(p. 708), `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller`  
(p. 782), `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` (p. 885),  
`activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 917), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1314),  
`activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1347),  
`activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` (p. 1379),  
`activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1411),  
`activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1458),  
`activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` (p. 1488),  
`activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1521),  
`activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1551),  
`activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1586),  
`activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1654),  
`activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1796),  
`activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1830),  
`activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1917),  
`activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 2017),  
`activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2178),  
`activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 2246),  
`activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 2275),  
`activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` (p. 2298),  
`activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller` (p. 2330),  
`activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2354),  
`activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller`  
(p. 2393), `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller`  
(p. 2444), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller`  
(p. 2664), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller`  
(p. 2706), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`  
(p. 2736), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2762),  
`activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2796), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2844), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2899),



activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 3026),  
activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 3135),  
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 3168),  
activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 3193),  
activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 3312),  
activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller  
(p. 3342), activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller  
(p. 3350), activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 3392),  
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 3487), ac-  
tivemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 3532), ac-  
tivemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3600),  
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3801),  
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller (p. 3954),  
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3984),  
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 4119),  
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 4158),  
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller  
(p. 209), activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller  
(p. 250), activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller  
(p. 338), activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller  
(p. 380), activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller  
(p. 408), activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller  
(p. 456), activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller  
(p. 502), activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller  
(p. 568), activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller  
(p. 597), activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller  
(p. 626), activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller  
(p. 656), activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller  
(p. 687), activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller  
(p. 717), activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller  
(p. 796), activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (p. 893),  
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (p. 926), ac-  
tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (p. 1322),  
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1356),  
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (p. 1387),  
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1419),  
activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1466),  
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (p. 1496),  
activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1529),  
activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1559),  
activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1594),  
activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1637),  
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1808),  
activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (p. 1838),  
activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1913),  
activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 2026),  
activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 2187),  
activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (p. 2238),  
activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (p. 2259),  
activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (p. 2306),

`activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 2326),  
`activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 2358),  
`activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller`  
(p. 2389), `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller`  
(p. 2440), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller`  
(p. 2672), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller`  
(p. 2702), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller`  
(p. 2744), `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` (p. 2766),  
`activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2783), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2835), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2891),  
`activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 3017),  
`activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 3144),  
`activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 3176),  
`activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 3206),  
`activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3307),  
`activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller`  
(p. 3338), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller`  
(p. 3371), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3402),  
`activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 3495), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 3515), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 3591),  
`activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller` (p. 3797),  
`activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3938),  
`activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3967),  
`activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller` (p. 4107),  
`activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 4171),  
`activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller`  
(p. 213), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller`  
(p. 254), `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller`  
(p. 346), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller`  
(p. 388), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller`  
(p. 417), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller`  
(p. 464), `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller`  
(p. 511), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller`  
(p. 577), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller`  
(p. 605), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller`  
(p. 635), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller`  
(p. 665), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller`  
(p. 691), `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller`  
(p. 721), `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller`  
(p. 803), `activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller` (p. 897),  
`activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 930), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1326),  
`activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1360),  
`activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller` (p. 1391),  
`activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1424),  
`activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1471),  
`activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller` (p. 1500),  
`activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1534),

activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller (p. 1563),  
 activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller (p. 1599),  
 activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller (p. 1642),  
 activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller (p. 1804),  
 activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller (p. 1818),  
 activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller (p. 1900),  
 activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller (p. 2004),  
 activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller (p. 2165),  
 activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller (p. 2230),  
 activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller (p. 2271),  
 activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller (p. 2294),  
 activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller (p. 2314),  
 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller (p. 2341),  
 activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller  
 (p. 2376), activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller  
 (p. 2427), activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller  
 (p. 2651), activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller  
 (p. 2714), activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller  
 (p. 2723), activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller (p. 2774),  
 activemq::wireformat::openwire::marshal::v6::MessageMarshaller (p. 2805), ac-  
 tivemq::wireformat::openwire::marshal::v6::MessagePullMarshaller (p. 2852), ac-  
 tivemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller (p. 2887),  
 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller (p. 3008),  
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller (p. 3148),  
 activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller (p. 3180),  
 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller (p. 3214),  
 activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller (p. 3295),  
 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller  
 (p. 3333), activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller  
 (p. 3367), activemq::wireformat::openwire::marshal::v6::ResponseMarshaller (p. 3417),  
 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller (p. 3491), ac-  
 tivemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller (p. 3511), ac-  
 tivemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller (p. 3579),  
 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller (p. 3805),  
 activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller (p. 3958),  
 activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller (p. 3979),  
 activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller (p. 4111),  
 and activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller (p. 4150).

6.301.3.6 virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::tightMarshal2  
 ( OpenWireFormat \* *format*, commands::DataStructure \* *command*,  
 decaf::io::DataOutputStream \* *ds*, utils::BooleanStream \* *bs* ) throw (   
 decaf::io::IOException ) [pure virtual]

Tight Marshal to the given stream.

#### Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal

<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 197), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 242), `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 330), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 372), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 400), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 448), `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 494), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 560), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 589), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 618), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 653), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 683), `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 713), `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 790), `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller` (p. 889), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 922), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1318), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1352), `activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller` (p. 1384), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1416), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1463), `activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller` (p. 1492), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1526), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1555), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1591), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1659), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1800), `activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller` (p. 1834), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1922), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 2022), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2183), `activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller` (p. 2251), `activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller` (p. 2280), `activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller` (p. 2303), `activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller` (p. 2334), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2363), `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 2398), `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller` (p. 2449), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2669), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2711), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller` (p. 2741), `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller` (p. 2778),

activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2801), activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2849), activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2904), activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 3031), activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 3153), activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 3184), activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 3202), activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 3304), activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller (p. 3321), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller (p. 3354), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 3412), activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 3504), activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 3520), activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 3588), activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 3794), activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3942), activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3971), activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 4124), activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 4163), activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller (p. 205), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller (p. 259), activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller (p. 342), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller (p. 385), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller (p. 413), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller (p. 461), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller (p. 507), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller (p. 573), activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller (p. 601), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller (p. 631), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller (p. 661), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller (p. 696), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller (p. 726), activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller (p. 811), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 901), activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 935), activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1331), activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1339), activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1372), activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1403), activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1450), activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1480), activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1513), activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1543), activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1578), activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1646), activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1788), activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1822), activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1905), activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 2009), activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 2170),

`activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller` (p. 2235),  
`activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller` (p. 2264),  
`activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller` (p. 2287),  
`activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller` (p. 2318),  
`activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 2346),  
`activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller`  
(p. 2385), `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller`  
(p. 2432), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller`  
(p. 2656), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller`  
(p. 2694), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller`  
(p. 2728), `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller` (p. 2758),  
`activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2792), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2832), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller` (p. 2884),  
`activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 3013),  
`activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 3132),  
`activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller` (p. 3164),  
`activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 3198),  
`activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 3291),  
`activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller`  
(p. 3330), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller`  
(p. 3359), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3397),  
`activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller` (p. 3484), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 3529), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 3583),  
`activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller` (p. 3810),  
`activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3946),  
`activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3988),  
`activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller` (p. 4116),  
`activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 4154),  
`activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller`  
(p. 193), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller`  
(p. 238), `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller`  
(p. 326), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller`  
(p. 368), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller`  
(p. 396), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller`  
(p. 444), `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller`  
(p. 490), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller`  
(p. 556), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller`  
(p. 585), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller`  
(p. 614), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller`  
(p. 644), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller`  
(p. 675), `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller`  
(p. 704), `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller`  
(p. 776), `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller` (p. 881),  
`activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 914), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1310),  
`activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1343),  
`activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller` (p. 1376),  
`activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1407),

activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1454),  
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1484),  
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1517),  
activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1547),  
activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1582),  
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1651),  
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1792),  
activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1826),  
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1909),  
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 2013),  
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 2174),  
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 2243),  
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 2268),  
activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 2291),  
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 2322),  
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 2350),  
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller  
(p. 2381), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller  
(p. 2436), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller  
(p. 2660), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller  
(p. 2698), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller  
(p. 2732), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2770),  
activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2788), ac-  
tivismq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2840), ac-  
tivismq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2896),  
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 3022),  
activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 3140),  
activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 3172),  
activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 3210),  
activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 3299),  
activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller  
(p. 3325), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller  
(p. 3363), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 3407),  
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 3500), ac-  
tivismq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 3524), ac-  
tivismq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 3596),  
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 3790),  
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (p. 3950),  
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3976),  
activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 4128),  
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 4167),  
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller  
(p. 201), activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller  
(p. 246), activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller  
(p. 334), activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller  
(p. 376), activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller  
(p. 404), activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller  
(p. 452), activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller  
(p. 498), activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller  
(p. 565), activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller

(p. 593), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller`  
(p. 623), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller`  
(p. 648), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`  
(p. 679), `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller`  
(p. 709), `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller`  
(p. 783), `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` (p. 885),  
`activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 918), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1314),  
`activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1348),  
`activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` (p. 1380),  
`activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1411),  
`activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1458),  
`activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` (p. 1488),  
`activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1521),  
`activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1551),  
`activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1586),  
`activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1655),  
`activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1796),  
`activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1830),  
`activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1918),  
`activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 2018),  
`activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2179),  
`activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 2247),  
`activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 2276),  
`activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` (p. 2299),  
`activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller` (p. 2330),  
`activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2354),  
`activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller`  
(p. 2394), `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller`  
(p. 2445), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller`  
(p. 2664), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller`  
(p. 2706), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`  
(p. 2736), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2762),  
`activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2797), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2844), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2900),  
`activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 3027),  
`activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 3136),  
`activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller` (p. 3168),  
`activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 3193),  
`activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 3312),  
`activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller`  
(p. 3342), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller`  
(p. 3350), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3392),  
`activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller` (p. 3488), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 3533), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 3600),  
`activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller` (p. 3802),  
`activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3954),  
`activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3984),



activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 4120),  
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 4159),  
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller  
(p. 210), activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller  
(p. 251), activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller  
(p. 338), activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller  
(p. 380), activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller  
(p. 409), activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller  
(p. 456), activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller  
(p. 503), activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller  
(p. 569), activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller  
(p. 597), activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller  
(p. 627), activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller  
(p. 657), activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller  
(p. 687), activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller  
(p. 717), activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller  
(p. 797), activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (p. 893),  
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (p. 926), ac-  
tivismq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (p. 1323),  
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1356),  
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (p. 1388),  
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1420),  
activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1467),  
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (p. 1496),  
activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1530),  
activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1560),  
activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1595),  
activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1638),  
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1809),  
activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (p. 1838),  
activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1914),  
activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 2026),  
activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 2187),  
activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (p. 2239),  
activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (p. 2260),  
activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (p. 2307),  
activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (p. 2326),  
activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 2358),  
activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller  
(p. 2390), activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller  
(p. 2441), activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller  
(p. 2673), activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller  
(p. 2702), activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller  
(p. 2745), activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (p. 2766),  
activemq::wireformat::openwire::marshal::v5::MessageMarshaller (p. 2783), ac-  
tivismq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2836), ac-  
tivismq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (p. 2892),  
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (p. 3018),  
activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 3144),  
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (p. 3176),

`activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 3206),  
`activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3308),  
`activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller`  
(p. 3338), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller`  
(p. 3371), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3402),  
`activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 3496), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 3516), `ac-`  
`tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 3592),  
`activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller` (p. 3798),  
`activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3938),  
`activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3967),  
`activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller` (p. 4108),  
`activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 4171),  
`activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller`  
(p. 214), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller`  
(p. 255), `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller`  
(p. 346), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller`  
(p. 389), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller`  
(p. 417), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller`  
(p. 465), `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller`  
(p. 511), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller`  
(p. 577), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller`  
(p. 605), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller`  
(p. 635), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller`  
(p. 665), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller`  
(p. 692), `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller`  
(p. 721), `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller`  
(p. 804), `activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller` (p. 897),  
`activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 931), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1327),  
`activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1360),  
`activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller` (p. 1392),  
`activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1424),  
`activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1471),  
`activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller` (p. 1500),  
`activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1534),  
`activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1564),  
`activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1599),  
`activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1642),  
`activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1805),  
`activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller` (p. 1818),  
`activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1901),  
`activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 2005),  
`activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2166),  
`activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller` (p. 2231),  
`activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller` (p. 2272),  
`activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller` (p. 2295),  
`activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller` (p. 2314),  
`activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2341),  
`activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller`

(p. 2377), `activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller` (p. 2428), `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller` (p. 2652), `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller` (p. 2715), `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller` (p. 2724), `activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller` (p. 2774), `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2806), `activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2853), `activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller` (p. 2888), `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 3009), `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 3149), `activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller` (p. 3180), `activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 3215), `activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 3295), `activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller` (p. 3334), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller` (p. 3367), `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3417), `activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller` (p. 3492), `activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller` (p. 3512), `activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller` (p. 3579), `activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller` (p. 3806), `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3958), `activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3980), `activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller` (p. 4112), and `activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller` (p. 4150).

6.301.3.7 `virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::tightUnmarshal ( OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [pure virtual]

Tight Un-marshal to the given stream.

#### Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 197), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 243), `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 331), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 372), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 401), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 448), `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller`

(p. 495), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller`  
(p. 561), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller`  
(p. 590), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller`  
(p. 619), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller`  
(p. 653), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller`  
(p. 684), `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller`  
(p. 713), `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller`  
(p. 792), `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller` (p. 890),  
`activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 923), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1319),  
`activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1352),  
`activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller` (p. 1384),  
`activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1416),  
`activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1463),  
`activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller` (p. 1493),  
`activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1526),  
`activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1556),  
`activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1591),  
`activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1660),  
`activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1801),  
`activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller` (p. 1835),  
`activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1923),  
`activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 2022),  
`activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2183),  
`activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller` (p. 2251),  
`activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller` (p. 2280),  
`activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller` (p. 2303),  
`activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller` (p. 2335),  
`activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2363),  
`activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller`  
(p. 2399), `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller`  
(p. 2450), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller`  
(p. 2669), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller`  
(p. 2711), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller`  
(p. 2741), `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller` (p. 2779),  
`activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2802), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2849), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller` (p. 2904),  
`activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 3032),  
`activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 3153),  
`activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller` (p. 3185),  
`activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 3202),  
`activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3304),  
`activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller`  
(p. 3322), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller`  
(p. 3355), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3413),  
`activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller` (p. 3504), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 3521), `ac-`  
`tivemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3588),  
`activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller` (p. 3794),

activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3943),  
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3972),  
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 4124),  
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 4163),  
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller  
(p. 206), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller  
(p. 260), activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller  
(p. 343), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller  
(p. 385), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller  
(p. 413), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller  
(p. 461), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller  
(p. 507), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller  
(p. 574), activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller  
(p. 602), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller  
(p. 632), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller  
(p. 662), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller  
(p. 696), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller  
(p. 726), activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller  
(p. 813), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 902),  
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 935), ac-  
tivismq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1332),  
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1340),  
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1372),  
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1403),  
activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1450),  
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1480),  
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1513),  
activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1543),  
activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1578),  
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1647),  
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1788),  
activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1823),  
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1906),  
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 2010),  
activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 2170),  
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 2235),  
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 2264),  
activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 2287),  
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 2319),  
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 2346),  
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller  
(p. 2386), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller  
(p. 2433), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller  
(p. 2656), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller  
(p. 2694), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller  
(p. 2728), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2759),  
activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2793), ac-  
tivismq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2832), ac-  
tivismq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2884),  
activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 3014),

`activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 3132),  
`activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller` (p. 3165),  
`activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 3198),  
`activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 3291),  
`activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller`  
(p. 3330), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller`  
(p. 3359), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3398),  
`activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller` (p. 3484), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 3529), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 3584),  
`activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller` (p. 3810),  
`activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3947),  
`activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3989),  
`activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller` (p. 4116),  
`activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 4155),  
`activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller`  
(p. 193), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller`  
(p. 238), `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller`  
(p. 327), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller`  
(p. 368), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller`  
(p. 396), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller`  
(p. 444), `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller`  
(p. 490), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller`  
(p. 557), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller`  
(p. 586), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller`  
(p. 615), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller`  
(p. 645), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller`  
(p. 675), `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller`  
(p. 705), `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller`  
(p. 777), `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller` (p. 881),  
`activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 914), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1310),  
`activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1344),  
`activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller` (p. 1376),  
`activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1408),  
`activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1455),  
`activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller` (p. 1485),  
`activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1518),  
`activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1547),  
`activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1582),  
`activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1651),  
`activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1792),  
`activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller` (p. 1827),  
`activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1910),  
`activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 2014),  
`activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2175),  
`activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller` (p. 2243),  
`activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller` (p. 2268),  
`activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller` (p. 2291),  
`activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller` (p. 2323),

activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 2350),  
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller  
(p. 2382), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller  
(p. 2437), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller  
(p. 2661), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller  
(p. 2698), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller  
(p. 2733), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2771),  
activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2788), ac-  
tivismq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2841), ac-  
tivismq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2896),  
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 3023),  
activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 3141),  
activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 3173),  
activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 3211),  
activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 3300),  
activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller  
(p. 3326), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller  
(p. 3363), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 3408),  
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 3500), ac-  
tivismq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 3525), ac-  
tivismq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 3597),  
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 3790),  
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (p. 3951),  
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3976),  
activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 4128),  
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 4168),  
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller  
(p. 202), activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller  
(p. 247), activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller  
(p. 335), activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller  
(p. 377), activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller  
(p. 405), activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller  
(p. 453), activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller  
(p. 499), activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller  
(p. 565), activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller  
(p. 594), activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller  
(p. 623), activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller  
(p. 649), activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller  
(p. 679), activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller  
(p. 709), activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller  
(p. 784), activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller (p. 886),  
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller (p. 918), ac-  
tivismq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (p. 1315),  
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (p. 1348),  
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller (p. 1380),  
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (p. 1412),  
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1459),  
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (p. 1489),  
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1522),  
activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1552),

`activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1587),  
`activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1655),  
`activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1797),  
`activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1831),  
`activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1918),  
`activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 2018),  
`activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2179),  
`activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 2247),  
`activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 2276),  
`activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` (p. 2299),  
`activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller` (p. 2331),  
`activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2355),  
`activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller`  
(p. 2394), `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller`  
(p. 2445), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller`  
(p. 2665), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller`  
(p. 2707), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`  
(p. 2737), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2763),  
`activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2797), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2845), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2900),  
`activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 3027),  
`activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 3136),  
`activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller` (p. 3169),  
`activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 3194),  
`activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 3313),  
`activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller`  
(p. 3343), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller`  
(p. 3351), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3393),  
`activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller` (p. 3488), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 3533), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 3601),  
`activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller` (p. 3802),  
`activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3955),  
`activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3985),  
`activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller` (p. 4120),  
`activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 4159),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller`  
(p. 210), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`  
(p. 251), `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller`  
(p. 339), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller`  
(p. 381), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller`  
(p. 409), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller`  
(p. 457), `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller`  
(p. 503), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller`  
(p. 569), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller`  
(p. 598), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller`  
(p. 627), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller`  
(p. 657), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`  
(p. 688), `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller`



(p. 718), `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 799), `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` (p. 894), `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 927), `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1323), `activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1357), `activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1388), `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1420), `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1467), `activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1497), `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1530), `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1560), `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1595), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1638), `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1809), `activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1839), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1914), `activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 2027), `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2188), `activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` (p. 2239), `activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller` (p. 2260), `activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` (p. 2307), `activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 2327), `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 2359), `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 2390), `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 2441), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller` (p. 2673), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller` (p. 2703), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller` (p. 2745), `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` (p. 2767), `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2784), `activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2836), `activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2892), `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 3018), `activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 3145), `activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 3177), `activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 3207), `activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3308), `activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller` (p. 3339), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller` (p. 3372), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3403), `activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 3496), `activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 3516), `activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 3592), `activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller` (p. 3798), `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3939), `activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3968), `activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller` (p. 4108), `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 4172), `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 214), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller`

(p. 255), `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller`  
(p. 347), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller`  
(p. 389), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller`  
(p. 418), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller`  
(p. 465), `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller`  
(p. 512), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller`  
(p. 578), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller`  
(p. 606), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller`  
(p. 636), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller`  
(p. 666), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller`  
(p. 692), `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller`  
(p. 722), `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller`  
(p. 806), `activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller` (p. 898),  
`activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 931), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1327),  
`activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1361),  
`activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller` (p. 1392),  
`activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1425),  
`activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1472),  
`activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller` (p. 1501),  
`activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1535),  
`activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1564),  
`activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1600),  
`activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1643),  
`activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1805),  
`activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller` (p. 1819),  
`activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1901),  
`activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 2005),  
`activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2166),  
`activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller` (p. 2231),  
`activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller` (p. 2272),  
`activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller` (p. 2295),  
`activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller` (p. 2315),  
`activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2342),  
`activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller`  
(p. 2377), `activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller`  
(p. 2428), `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller`  
(p. 2652), `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller`  
(p. 2715), `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller`  
(p. 2724), `activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller` (p. 2775),  
`activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2806), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2853), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller` (p. 2888),  
`activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 3009),  
`activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 3149),  
`activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller` (p. 3181),  
`activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 3215),  
`activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 3296),  
`activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller`  
(p. 3334), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller`

(p. 3368), **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3418), **activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller** (p. 3492), **activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller** (p. 3512), **activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller** (p. 3580), **activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller** (p. 3806), **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** (p. 3959), **activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller** (p. 3980), **activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller** (p. 4112), and **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller** (p. 4151).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h`

## 6.302 activemq::commands::DataStructure Class Reference

```
#include <src/main/activemq/commands/DataStructure.h>
```

Inheritance diagram for `activemq::commands::DataStructure`:

### Public Member Functions

- virtual `~DataStructure ()`
- virtual unsigned char `getDataStructureType ()` const =0  
*Get the **DataStructure** (p. 1713) Type as defined in `CommandTypes.h`.*
- virtual `DataStructure * cloneDataStructure ()` const =0  
*Clone this obbjet and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void `copyDataStructure (const DataStructure *src)=0`  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string `toString ()` const =0  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool `equals (const DataStructure *value)` const =0  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*

### 6.302.1 Constructor & Destructor Documentation

6.302.1.1 `virtual activemq::commands::DataStructure::~DataStructure ( ) [inline, virtual]`

### 6.302.2 Member Function Documentation

6.302.2.1 `virtual DataStructure* activemq::commands::DataStructure::cloneDataStructure ( ) const [pure virtual]`

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implemented in `activemq::commands::ActiveMQBlobMessage` (p. 186), `activemq::commands::ActiveMQCommand` (p. 219), `activemq::commands::ActiveMQDestination` (p. 315), `activemq::commands::ActiveMQMapMessage` (p. 354), `activemq::commands::ActiveMQMessage` (p. 391), `activemq::commands::ActiveMQObjectMessage` (p. 439), `activemq::commands::ActiveMQQueue` (p. 481), `activemq::commands::ActiveMQStreamMessage` (p. 539), `activemq::commands::ActiveMQTempDestination` (p. 580), `activemq::commands::ActiveMQTempMessage` (p. 608), `activemq::commands::ActiveMQTempTopic` (p. 638), `activemq::commands::ActiveMQTextMessage` (p. 669), `activemq::commands::ActiveMQTopic` (p. 698), `activemq::commands::BooleanExpression` (p. 862), `activemq::commands::BrokerError` (p. 870), `activemq::commands::BrokerId` (p. 876), `activemq::commands::BrokerInfo` (p. 904), `activemq::commands::ConnectionControl` (p. 1303), `activemq::commands::ConnectionError` (p. 1333), `activemq::commands::ConnectionId` (p. 1366), `activemq::commands::ConnectionInfo` (p. 1395), `activemq::commands::ConsumerControl` (p. 1443), `activemq::commands::ConsumerId` (p. 1474), `activemq::commands::ConsumerInfo` (p. 1504), `activemq::commands::ControlCommand` (p. 1537), `activemq::commands::DataArrayResponse` (p. 1573), `activemq::commands::DataResponse` (p. 1633), `activemq::commands::DestinationInfo` (p. 1781), `activemq::commands::DiscoveryEvent` (p. 1813), `activemq::commands::ExceptionResponse` (p. 1896), `activemq::commands::FlushCommand` (p. 2000), `activemq::commands::IntegerResponse` (p. 2161), `activemq::commands::JournalQueueAck` (p. 2225), `activemq::commands::JournalTopicAck` (p. 2253), `activemq::commands::JournalTrace` (p. 2282), `activemq::commands::JournalTransaction` (p. 2309), `activemq::commands::KeepAliveInfo` (p. 2336), `activemq::commands::LastPartialCommand` (p. 2372), `activemq::commands::LocalTransactionId` (p. 2422), `activemq::commands::Message` (p. 2601), `activemq::commands::MessageAck` (p. 2644), `activemq::commands::MessageDispatch` (p. 2680), `activemq::commands::MessageDispatchNotification` (p. 2717), `activemq::commands::MessageId` (p. 2752), `activemq::commands::MessagePull` (p. 2825), `activemq::commands::NetworkBridgeFilter` (p. 2878), `activemq::commands::PartialCommand` (p. 3003), `activemq::commands::ProducerAck` (p. 3126), `activemq::commands::ProducerId` (p. 3158), `activemq::commands::ProducerInfo` (p. 3187), `activemq::commands::RemoveInfo` (p. 3285), `activemq::commands::RemoveSubscriptionInfo` (p. 3315), `activemq::commands::ReplayCommand` (p. 3345), `activemq::commands::Response` (p. 3380), `activemq::commands::SessionId` (p. 3478), `activemq::commands::SessionInfo` (p. 3506), `activemq::commands::ShutdownInfo` (p. 3574), `activemq::commands::SubscriptionInfo` (p. 3783), `activemq::commands::TransactionId` (p. 3933), `activemq::commands::TransactionInfo` (p. 3961), `activemq::commands::WireFormatInfo` (p. 4097), and `activemq::commands::XATransactionId` (p. 4144).

**6.302.2.2** `virtual void activemq::commands::DataStructure::copyDataStructure ( const DataStructure * src ) [pure virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

### Returns

src - Source Object

Implemented in `activemq::commands::ActiveMQBlobMessage` (p. 186), `activemq::commands::ActiveMQBytesMessage` (p. 219), `activemq::commands::ActiveMQDestination` (p. 316), `activemq::commands::ActiveMQMapMessage` (p. 354), `activemq::commands::ActiveMQMessage` (p. 391), `activemq::commands::ActiveMQObjectMessage` (p. 439), `activemq::commands::ActiveMQQueue` (p. 481), `activemq::commands::ActiveMQStreamMessage` (p. 540), `activemq::commands::ActiveMQTempDestination` (p. 580), `activemq::commands::ActiveMQTempQueue` (p. 609), `activemq::commands::ActiveMQTempTopic` (p. 639), `activemq::commands::ActiveMQTextMessage` (p. 669), `activemq::commands::ActiveMQTopic` (p. 699), `activemq::commands::BaseCommand` (p. 766), `activemq::commands::BrokerError` (p. 870), `activemq::commands::BrokerId` (p. 876), `activemq::commands::BrokerInfo` (p. 904), `activemq::commands::ConnectionControl` (p. 1303), `activemq::commands::ConnectionError` (p. 1334), `activemq::commands::ConnectionId` (p. 1367), `activemq::commands::ConnectionInfo` (p. 1395), `activemq::commands::ConsumerControl` (p. 1443), `activemq::commands::ConsumerId` (p. 1474), `activemq::commands::ConsumerInfo` (p. 1504), `activemq::commands::ControlCommand` (p. 1537), `activemq::commands::DataArrayResponse` (p. 1573), `activemq::commands::DataResponse` (p. 1633), `activemq::commands::DestinationInfo` (p. 1781), `activemq::commands::DiscoveryEvent` (p. 1813), `activemq::commands::ExceptionResponse` (p. 1896), `activemq::commands::FlushCommand` (p. 2000), `activemq::commands::IntegerResponse` (p. 2161), `activemq::commands::JournalQueueAck` (p. 2225), `activemq::commands::JournalTopicAck` (p. 2253), `activemq::commands::JournalTrace` (p. 2282), `activemq::commands::JournalTransaction` (p. 2309), `activemq::commands::KeepAliveInfo` (p. 2336), `activemq::commands::LastPartialCommand` (p. 2372), `activemq::commands::LocalTransactionId` (p. 2422), `activemq::commands::Message` (p. 2602), `activemq::commands::MessageAck` (p. 2645), `activemq::commands::MessageDispatch` (p. 2680), `activemq::commands::MessageDispatchNotification` (p. 2718), `activemq::commands::MessageId` (p. 2752), `activemq::commands::MessagePull` (p. 2826), `activemq::commands::NetworkBridgeFilter` (p. 2878), `activemq::commands::PartialCommand` (p. 3003), `activemq::commands::ProducerAck` (p. 3126), `activemq::commands::ProducerId` (p. 3159), `activemq::commands::ProducerInfo` (p. 3187), `activemq::commands::RemoveInfo` (p. 3285), `activemq::commands::RemoveSubscriptionInfo` (p. 3315), `activemq::commands::ReplayCommand` (p. 3345), `activemq::commands::Response` (p. 3380), `activemq::commands::SessionId` (p. 3478), `activemq::commands::SessionInfo` (p. 3506), `activemq::commands::ShutdownInfo` (p. 3574), `activemq::commands::SubscriptionInfo` (p. 3784), `activemq::commands::TransactionId` (p. 3934), `activemq::commands::TransactionInfo` (p. 3961), `activemq::commands::WireFormatInfo` (p. 4097), and `activemq::commands::XATransactionId` (p. 4145).

**6.302.2.3** `virtual bool activemq::commands::DataStructure::equals ( const DataStructure * value ) const [pure virtual]`

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.



## Returns

The type of the data structure

Implemented in `activemq::commands::ActiveMQBlobMessage` (p. 187), `activemq::commands::ActiveMQBytesMessage` (p. 221), `activemq::commands::ActiveMQDestination` (p. 317), `activemq::commands::ActiveMQMapMessage` (p. 356), `activemq::commands::ActiveMQMessage` (p. 392), `activemq::commands::ActiveMQObjectMessage` (p. 440), `activemq::commands::ActiveMQQueue` (p. 482), `activemq::commands::ActiveMQStreamMessage` (p. 540), `activemq::commands::ActiveMQTempDestination` (p. 581), `activemq::commands::ActiveMQTempQueue` (p. 610), `activemq::commands::ActiveMQTempTopic` (p. 640), `activemq::commands::ActiveMQTextMessage` (p. 669), `activemq::commands::ActiveMQTopic` (p. 700), `activemq::commands::BrokerError` (p. 871), `activemq::commands::BrokerId` (p. 877), `activemq::commands::BrokerInfo` (p. 906), `activemq::commands::ConnectionControl` (p. 1304), `activemq::commands::ConnectionError` (p. 1334), `activemq::commands::ConnectionId` (p. 1367), `activemq::commands::ConnectionInfo` (p. 1396), `activemq::commands::ConsumerControl` (p. 1444), `activemq::commands::ConsumerId` (p. 1475), `activemq::commands::ConsumerInfo` (p. 1505), `activemq::commands::ControlCommand` (p. 1538), `activemq::commands::DataArrayResponse` (p. 1574), `activemq::commands::DataResponse` (p. 1634), `activemq::commands::DestinationInfo` (p. 1782), `activemq::commands::DiscoveryEvent` (p. 1814), `activemq::commands::ExceptionResponse` (p. 1897), `activemq::commands::FlushCommand` (p. 2001), `activemq::commands::IntegerResponse` (p. 2162), `activemq::commands::JournalQueueAck` (p. 2226), `activemq::commands::JournalTopicAck` (p. 2254), `activemq::commands::JournalTrace` (p. 2283), `activemq::commands::JournalTransaction` (p. 2310), `activemq::commands::KeepAliveInfo` (p. 2337), `activemq::commands::LastPartialCommand` (p. 2373), `activemq::commands::LocalTransactionId` (p. 2423), `activemq::commands::Message` (p. 2604), `activemq::commands::MessageAck` (p. 2645), `activemq::commands::MessageDispatch` (p. 2681), `activemq::commands::MessageDispatchNotification` (p. 2718), `activemq::commands::MessageId` (p. 2753), `activemq::commands::MessagePull` (p. 2826), `activemq::commands::NetworkBridgeFilter` (p. 2879), `activemq::commands::PartialCommand` (p. 3004), `activemq::commands::ProducerAck` (p. 3127), `activemq::commands::ProducerId` (p. 3159), `activemq::commands::ProducerInfo` (p. 3188), `activemq::commands::RemoveInfo` (p. 3286), `activemq::commands::RemoveSubscriptionInfo` (p. 3316), `activemq::commands::ReplayCommand` (p. 3346), `activemq::commands::Response` (p. 3381), `activemq::commands::SessionId` (p. 3479), `activemq::commands::SessionInfo` (p. 3507), `activemq::commands::ShutdownInfo` (p. 3575), `activemq::commands::SubscriptionInfo` (p. 3784), `activemq::commands::TransactionId` (p. 3934), `activemq::commands::TransactionInfo` (p. 3962), `activemq::commands::WireFormatInfo` (p. 4098), and `activemq::commands::XATransactionId` (p. 4145).

**6.302.2.5** `virtual std::string activemq::commands::DataStructure::toString ( ) const` [pure virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

## Returns

formatted string useful for debugging.

Implemented in `activemq::commands::ActiveMQBlobMessage` (p. 189), `activemq::commands::ActiveMQBytesMessage` (p. 229), `activemq::commands::ActiveMQDestination` (p. 321), `activemq::commands::ActiveMQMapMessage` (p. 364), `activemq::commands::ActiveMQMessage` (p. 392), `activemq::commands::ActiveMQObjectMessage` (p. 440), `activemq::commands::ActiveMQQueue` (p. 483), `activemq::commands::ActiveMQStreamMessage`

(p. 547), `activemq::commands::ActiveMQTempDestination` (p. 582), `activemq::commands::ActiveMQTempTopic` (p. 611), `activemq::commands::ActiveMQTextMessage` (p. 671), `activemq::commands::ActiveMQTopic` (p. 701), `activemq::commands::BaseCommand` (p. 770), `activemq::commands::BaseDataStructure` (p. 841), `activemq::commands::BooleanExpression` (p. 863), `activemq::commands::BrokerId` (p. 877), `activemq::commands::BrokerInfo` (p. 908), `activemq::commands::Command` (p. 1231), `activemq::commands::ConnectionControl` (p. 1305), `activemq::commands::ConnectionError` (p. 1335), `activemq::commands::ConnectionId` (p. 1368), `activemq::commands::ConnectionInfo` (p. 1398), `activemq::commands::ConsumerControl` (p. 1445), `activemq::commands::ConsumerId` (p. 1476), `activemq::commands::ConsumerInfo` (p. 1507), `activemq::commands::ControlCommand` (p. 1538), `activemq::commands::DataArrayResponse` (p. 1574), `activemq::commands::DataResponse` (p. 1634), `activemq::commands::DestinationInfo` (p. 1783), `activemq::commands::DiscoveryEvent` (p. 1814), `activemq::commands::ExceptionResponse` (p. 1897), `activemq::commands::FlushCommand` (p. 2001), `activemq::commands::IntegerResponse` (p. 2162), `activemq::commands::JournalQueueAck` (p. 2227), `activemq::commands::JournalTopicAck` (p. 2255), `activemq::commands::JournalTrace` (p. 2283), `activemq::commands::JournalTransaction` (p. 2310), `activemq::commands::KeepAliveInfo` (p. 2337), `activemq::commands::LastPartialCommand` (p. 2373), `activemq::commands::LocalTransactionId` (p. 2424), `activemq::commands::Message` (p. 2611), `activemq::commands::MessageAck` (p. 2647), `activemq::commands::MessageDispatch` (p. 2682), `activemq::commands::MessageDispatchNotification` (p. 2719), `activemq::commands::MessageId` (p. 2754), `activemq::commands::MessagePull` (p. 2827), `activemq::commands::NetworkBridgeFilter` (p. 2880), `activemq::commands::PartialCommand` (p. 3005), `activemq::commands::ProducerAck` (p. 3127), `activemq::commands::ProducerId` (p. 3160), `activemq::commands::ProducerInfo` (p. 3189), `activemq::commands::RemoveInfo` (p. 3287), `activemq::commands::RemoveSubscriptionInfo` (p. 3317), `activemq::commands::ReplayCommand` (p. 3346), `activemq::commands::Response` (p. 3382), `activemq::commands::SessionId` (p. 3480), `activemq::commands::SessionInfo` (p. 3507), `activemq::commands::ShutdownInfo` (p. 3575), `activemq::commands::SubscriptionInfo` (p. 3785), `activemq::commands::TransactionId` (p. 3935), `activemq::commands::TransactionInfo` (p. 3963), `activemq::commands::WireFormatInfo` (p. 4104), and `activemq::commands::XATransactionId` (p. 4146).

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DataStructure.h`

### 6.303 decaf::util::Date Class Reference

Wrapper class around a time value in milliseconds.

```
#include <src/main/decaf/util/Date.h>
```

Inheritance diagram for `decaf::util::Date`:

#### Public Member Functions

- `Date ()`

*Default constructor - sets time to the current System time, rounded to the nearest millisecond.*



- **Date** (long long milliseconds)  
*Constructs the date with a given time value.*
- **Date** (const **Date** &source)  
*Copy constructor.*
- **Date & operator=** (const **Date** &value)  
*Assigns the value of one **Date** (p. 1719) object to another.*
- virtual ~**Date** ()
- long long **getTime** () const  
*Gets the underlying time.*
- void **setTime** (long long milliseconds)  
*Sets the underlying time.*
- bool **after** (const **Date** &when) const  
*Determines whether or not this date falls after the specified time.*
- bool **before** (const **Date** &when) const  
*Determines whether or not this date falls before the specified time.*
- std::string **toString** () const  
*Converts this **Date** (p. 1719) object to a String of the form:*
- virtual int **compareTo** (const **Date** &value) const  
*Compares this Data object to the one given.*
- virtual bool **equals** (const **Date** &value) const
- virtual bool **operator==** (const **Date** &value) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Date** &value) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*

### 6.303.1 Detailed Description

Wrapper class around a time value in milliseconds. This class is comparable to Java's java.util.Date class.

Since

1.0

### 6.303.2 Constructor & Destructor Documentation

#### 6.303.2.1 `decaf::util::Date::Date ( )`

Default constructor - sets time to the current System time, rounded to the nearest millisecond.

#### 6.303.2.2 `decaf::util::Date::Date ( long long milliseconds )`

Constructs the date with a given time value.

##### Parameters

<i>milliseconds</i>	The time in milliseconds;
---------------------	---------------------------

#### 6.303.2.3 `decaf::util::Date::Date ( const Date & source )`

Copy constructor.

##### Parameters

<i>source</i>	The <b>Date</b> (p. 1719) instance to copy into this one.
---------------	---

#### 6.303.2.4 `virtual decaf::util::Date::~~Date ( ) [virtual]`

### 6.303.3 Member Function Documentation

#### 6.303.3.1 `bool decaf::util::Date::after ( const Date & when ) const`

Determines whether or not this date falls after the specified time.

##### Parameters

<i>when</i>	The date to compare
-------------	---------------------

##### Returns

true if this date falls after when.

#### 6.303.3.2 `bool decaf::util::Date::before ( const Date & when ) const`

Determines whether or not this date falls before the specified time.

##### Parameters

<i>when</i>	The date to compare
-------------	---------------------

**Returns**

true if this date falls before when.

**6.303.3.3** `virtual int decaf::util::Date::compareTo ( const Date & value ) const` [virtual]

Compares this Date object to the one given.

**Parameters**

<i>value</i>	The <b>Date</b> (p. 1719) value to compare to this one.
--------------	---

**Returns**

zero if the **Date** (p. 1719) values are equal, a value less than zero if this Date value is earlier than argument value, and a value greater than zero if this **Date** (p. 1719) object is later than the argument **Date** (p. 1719) value.

**6.303.3.4** `virtual bool decaf::util::Date::equals ( const Date & value ) const` [virtual]

**Returns**

true if this value is considered equal to the passed value.

**6.303.3.5** `long long decaf::util::Date::getTime ( ) const`

Gets the underlying time.

**Returns**

The underlying time value in milliseconds.

**6.303.3.6** `virtual bool decaf::util::Date::operator< ( const Date & value ) const`  
[virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

**Parameters**

<i>value</i>	- the value to be compared to this one.
--------------	---

**Returns**

true if this object is equal to the one passed.

**6.303.3.7 Date& decaf::util::Date::operator= ( const Date & value )**

Assigns the value of one **Date** (p. 1719) object to another.

**Parameters**

<i>value</i>	The value to be copied into this <b>Date</b> (p. 1719) object.
--------------	--

**Returns**

reference to this object with the newly assigned value.

**6.303.3.8 virtual bool decaf::util::Date::operator== ( const Date & value ) const**  
[virtual]

Compares equality between this object and the one passed.

**Parameters**

<i>value</i>	- the value to be compared to this one.
--------------	---

**Returns**

true if this object is equal to the one passed.

**6.303.3.9 void decaf::util::Date::setTime ( long long milliseconds )**

Sets the underlying time.

**Parameters**

<i>milliseconds</i>	The underlying time value in milliseconds.
---------------------	--

**6.303.3.10 std::string decaf::util::Date::toString ( ) const**

Converts this **Date** (p. 1719) object to a String of the form:

dow mon dd hh:mm:ss zzz yyyy

where:

- dow is the day of the week (Sun, Mon, Tue, Wed, Thu, Fri, Sat).
- mon is the month (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec).
- dd is the day of the month (01 through 31), as two decimal digits.
- hh is the hour of the day (00 through 23), as two decimal digits.
- mm is the minute within the hour (00 through 59), as two decimal digits.

- ss is the second within the minute (00 through 61, as two decimal digits).
- zzz is the time zone (and may reflect daylight saving time). Standard time zone abbreviations include those recognized by the method parse. If time zone information is not available, then zzz is empty - that is, it consists of no characters at all.
- yyyy is the year, as four decimal digits.

**Returns**

the String representation of the **Date** (p. 1719) object.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Date.h**

**6.304 decaf::internal::DecafRuntime Class Reference**

Handles APR initialization and termination.

```
#include <src/main/decaf/internal/DecafRuntime.h>
```

Inheritance diagram for decaf::internal::DecafRuntime:

**Public Member Functions**

- **DecafRuntime ()**  
*Initializes the APR Runtime for a library.*
- virtual **~DecafRuntime ()**  
*Terminates the APR Runtime for a library.*
- apr\_pool\_t \* **getGlobalPool () const**  
*Grants access to the Global APR Pool instance that should be used when creating new threads.*

**6.304.1 Detailed Description**

Handles APR initialization and termination.

**6.304.2 Constructor & Destructor Documentation****6.304.2.1 decaf::internal::DecafRuntime::DecafRuntime ( )**

Initializes the APR Runtime for a library.

#### 6.304.2.2 `virtual decaf::internal::DecafRuntime::~~DecafRuntime ( ) [virtual]`

Terminates the APR Runtime for a library.

### 6.304.3 Member Function Documentation

#### 6.304.3.1 `apr_pool_t* decaf::internal::DecafRuntime::getGlobalPool ( ) const`

Grants access to the Global APR Pool instance that should be used when creating new threads.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/DecafRuntime.h`

## 6.305 `activemq::threads::DedicatedTaskRunner` Class Reference

```
#include <src/main/activemq/threads/DedicatedTaskRunner.h>
```

Inheritance diagram for `activemq::threads::DedicatedTaskRunner`:

### Public Member Functions

- **DedicatedTaskRunner** (**Task** \*task)
- virtual **~DedicatedTaskRunner** ()
- virtual void **shutdown** (unsigned int timeout)  
*Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.*
- virtual void **shutdown** ()  
*Shutdown once the task has finished and the TaskRunner's thread has exited.*
- virtual void **wakeup** ()  
*Signal the **TaskRunner** (p. 3849) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3846) instance will be run until its iterate method has returned false indicating it is done.*

### Protected Member Functions

- virtual void **run** ()  
*Run method - called by the Thread class in the context of the thread.*

### 6.305.1 Constructor & Destructor Documentation

6.305.1.1 **activemq::threads::DedicatedTaskRunner::DedicatedTaskRunner ( Task \* *task* )**

6.305.1.2 **virtual activemq::threads::DedicatedTaskRunner::~~DedicatedTaskRunner ( )**  
[virtual]

### 6.305.2 Member Function Documentation

6.305.2.1 **virtual void activemq::threads::DedicatedTaskRunner::run ( )** [protected,  
virtual]

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 3419).

6.305.2.2 **virtual void activemq::threads::DedicatedTaskRunner::shutdown ( )** [virtual]

Shutdown once the task has finished and the TaskRunner's thread has exited.

Implements **activemq::threads::TaskRunner** (p. 3849).

6.305.2.3 **virtual void activemq::threads::DedicatedTaskRunner::shutdown ( unsigned int  
*timeout* )** [virtual]

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

#### Parameters

<i>timeout</i>	- Time in Milliseconds to wait for the task to stop.
----------------	--

Implements **activemq::threads::TaskRunner** (p. 3849).

6.305.2.4 **virtual void activemq::threads::DedicatedTaskRunner::wakeup ( )** [virtual]

Signal the **TaskRunner** (p. 3849) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3846) instance will be run until its iterate method has returned false indicating it is done.

Implements **activemq::threads::TaskRunner** (p. 3850).

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**DedicatedTaskRunner.h**

## 6.306 activemq::core::policies::DefaultPrefetchPolicy Class Reference

```
#include <src/main/activemq/core/policies/DefaultPrefetchPolicy.h>
```

Inheritance diagram for activemq::core::policies::DefaultPrefetchPolicy:

### Public Member Functions

- **DefaultPrefetchPolicy ()**
- virtual **~DefaultPrefetchPolicy ()**
- virtual void **setDurableTopicPrefetch** (int value)  
*Sets the amount of prefetched messages for a Durable Topic.*
- virtual int **getDurableTopicPrefetch** () const  
*Gets the amount of messages to prefetch for a Durable Topic.*
- virtual void **setQueuePrefetch** (int value)  
*Sets the amount of prefetched messages for a Queue.*
- virtual int **getQueuePrefetch** () const  
*Gets the amount of messages to prefetch for a Queue.*
- virtual void **setQueueBrowserPrefetch** (int value)  
*Sets the amount of prefetched messages for a Queue Browser.*
- virtual int **getQueueBrowserPrefetch** () const  
*Gets the amount of messages to prefetch for a Queue Browser.*
- virtual void **setTopicPrefetch** (int value)  
*Sets the amount of prefetched messages for a Topic.*
- virtual int **getTopicPrefetch** () const  
*Gets the amount of messages to prefetch for a Topic.*
- virtual int **getMaxPrefetchLimit** (int value) const  
*Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.*
- virtual **PrefetchPolicy \* clone** () const  
*Clone the Policy and return a new pointer to that clone.*



## Static Public Attributes

- static int **MAX\_PREFETCH\_SIZE**
- static int **DEFAULT\_DURABLE\_TOPIC\_PREFETCH**
- static int **DEFAULT\_QUEUE\_PREFETCH**
- static int **DEFAULT\_QUEUE\_BROWSER\_PREFETCH**
- static int **DEFAULT\_TOPIC\_PREFETCH**

## 6.306.1 Constructor & Destructor Documentation

6.306.1.1 **activemq::core::policies::DefaultPrefetchPolicy::DefaultPrefetchPolicy ( )**

6.306.1.2 **virtual activemq::core::policies::DefaultPrefetchPolicy::~~DefaultPrefetchPolicy ( )**  
[virtual]

## 6.306.2 Member Function Documentation

6.306.2.1 **virtual PrefetchPolicy\* activemq::core::policies::DefaultPrefetchPolicy::clone ( )**  
**const** [virtual]

Clone the Policy and return a new pointer to that clone.

### Returns

pointer to a new **PrefetchPolicy** (p. 3062) instance that is a clone of this one.

Implements **activemq::core::PrefetchPolicy** (p. 3064).

6.306.2.2 **virtual int activemq::core::policies::DefaultPrefetchPolicy::getDurableTopicPrefetch ( ) const** [inline, virtual]

Gets the amount of messages to prefetch for a Durable Topic.

### Returns

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 3065).

6.306.2.3 **virtual int activemq::core::policies::DefaultPrefetchPolicy::getMaxPrefetchLimit ( int value ) const** [inline, virtual]

Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.

### Returns

the allowable value for a prefetch limit, either requested or the max.

Implements **activemq::core::PrefetchPolicy** (p. 3065).

**6.306.2.4** `virtual int activemq::core::policies::DefaultPrefetchPolicy::getQueueBrowserPrefetch ( ) const [inline, virtual]`

Gets the amount of messages to prefetch for a Queue Browser.

#### Returns

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 3065).

**6.306.2.5** `virtual int activemq::core::policies::DefaultPrefetchPolicy::getQueuePrefetch ( ) const [inline, virtual]`

Gets the amount of messages to prefetch for a Queue.

#### Returns

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 3065).

**6.306.2.6** `virtual int activemq::core::policies::DefaultPrefetchPolicy::getTopicPrefetch ( ) const [inline, virtual]`

Gets the amount of messages to prefetch for a Topic.

#### Returns

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 3066).

**6.306.2.7** `virtual void activemq::core::policies::DefaultPrefetchPolicy::setDurableTopicPrefetch ( int value ) [inline, virtual]`

Sets the amount of prefetched messages for a Durable Topic.

#### Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implements **activemq::core::PrefetchPolicy** (p. 3066).

**6.306.2.8** `virtual void activemq::core::policies::DefaultPrefetchPolicy::setQueueBrowserPrefetch ( int value ) [inline, virtual]`

Sets the amount of prefetched messages for a Queue Browser.

#### Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implements **activemq::core::PrefetchPolicy** (p. 3066).

**6.306.2.9** `virtual void activemq::core::policies::DefaultPrefetchPolicy::setQueuePrefetch ( int value ) [inline, virtual]`

Sets the amount of prefetched messages for a Queue.

#### Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implements **activemq::core::PrefetchPolicy** (p. 3066).

**6.306.2.10** `virtual void activemq::core::policies::DefaultPrefetchPolicy::setTopicPrefetch ( int value ) [inline, virtual]`

Sets the amount of prefetched messages for a Topic.

#### Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implements **activemq::core::PrefetchPolicy** (p. 3067).

### 6.306.3 Field Documentation

- 6.306.3.1 `int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT_DURABLE_TOPIC_PREFETCH` `[static]`
- 6.306.3.2 `int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT_QUEUE_BROWSER_PREFETCH` `[static]`
- 6.306.3.3 `int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT_QUEUE_PREFETCH` `[static]`
- 6.306.3.4 `int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT_TOPIC_PREFETCH` `[static]`
- 6.306.3.5 `int activemq::core::policies::DefaultPrefetchPolicy::MAX_PREFETCH_SIZE` `[static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/core/policies/DefaultPrefetchPolicy.h`

### 6.307 `activemq::core::policies::DefaultRedeliveryPolicy` Class Reference

```
#include <src/main/activemq/core/policies/DefaultRedeliveryPolicy.h>
```

Inheritance diagram for `activemq::core::policies::DefaultRedeliveryPolicy`:

#### Public Member Functions

- **DefaultRedeliveryPolicy** ()
- virtual **~DefaultRedeliveryPolicy** ()
- virtual double **getBackOffMultiplier** () const
- virtual void **setBackOffMultiplier** (double value)  
*Sets the Back-Off Multiplier for Message Redelivery.*
- virtual short **getCollisionAvoidancePercent** () const
- virtual void **setCollisionAvoidancePercent** (short value)
- virtual long long **getInitialRedeliveryDelay** () const  
*Gets the initial time that redelivery of messages is delayed.*
- virtual void **setInitialRedeliveryDelay** (long long value)  
*Sets the initial time that redelivery will be delayed.*

- virtual int **getMaximumRedeliveries** () const  
*Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.*
- virtual void **setMaximumRedeliveries** (int value)  
*Sets the Maximum allowable redeliveries for a Message.*
- virtual bool **isUseCollisionAvoidance** () const
- virtual void **setUseCollisionAvoidance** (bool value)
- virtual bool **isUseExponentialBackOff** () const
- virtual void **setUseExponentialBackOff** (bool value)
- virtual long long **getRedeliveryDelay** (long long previousDelay)  
*Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.*
- virtual **RedeliveryPolicy** \* **clone** () const  
*Create a copy of this Policy and return it.*

### 6.307.1 Constructor & Destructor Documentation

6.307.1.1 **activemq::core::policies::DefaultRedeliveryPolicy::DefaultRedeliveryPolicy** ( )

6.307.1.2 **virtual activemq::core::policies::DefaultRedeliveryPolicy::~~DefaultRedeliveryPolicy** ( ) [virtual]

### 6.307.2 Member Function Documentation

6.307.2.1 **virtual RedeliveryPolicy\* activemq::core::policies::DefaultRedeliveryPolicy::clone** ( ) const [virtual]

Create a copy of this Policy and return it.

#### Returns

pointer to a new **RedeliveryPolicy** (p. 3267) that is a copy of this one.

Implements **activemq::core::RedeliveryPolicy** (p. 3269).

6.307.2.2 **virtual double activemq::core::policies::DefaultRedeliveryPolicy::getBackOffMultiplier** ( ) const [inline, virtual]

#### Returns

The value of the Back-Off Multiplier for Message Redelivery.

Implements **activemq::core::RedeliveryPolicy** (p. 3270).

**6.307.2.3** `virtual short activemq::core::policies::DefaultRedeliveryPolicy::getCollisionAvoidancePercent ( ) const [virtual]`

#### Returns

the currently set Collision Avoidance percentage.

Implements **activemq::core::RedeliveryPolicy** (p. 3270).

**6.307.2.4** `virtual long long activemq::core::policies::DefaultRedeliveryPolicy::getInitialRedeliveryDelay ( ) const [inline, virtual]`

Gets the initial time that redelivery of messages is delayed.

#### Returns

the time in milliseconds that redelivery is delayed initially.

Implements **activemq::core::RedeliveryPolicy** (p. 3270).

**6.307.2.5** `virtual int activemq::core::policies::DefaultRedeliveryPolicy::getMaximumRedeliveries ( ) const [inline, virtual]`

Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.

#### Returns

maximum allowed redeliveries for a message.

Implements **activemq::core::RedeliveryPolicy** (p. 3270).

**6.307.2.6** `virtual long long activemq::core::policies::DefaultRedeliveryPolicy::getRedeliveryDelay ( long long previousDelay ) [virtual]`

Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.

#### Parameters

<i>previousDelay</i>	The last delay that was used between message redeliveries.
----------------------	--

#### Returns

the new delay to use before attempting another redelivery.

Implements **activemq::core::RedeliveryPolicy** (p. 3270).

**6.307.2.7** `virtual bool activemq::core::policies::DefaultRedeliveryPolicy::isUseCollisionAvoidance ( ) const [inline, virtual]`

#### Returns

whether or not collision avoidance is enabled for this Policy.

Implements **activemq::core::RedeliveryPolicy** (p. 3271).

**6.307.2.8** `virtual bool activemq::core::policies::DefaultRedeliveryPolicy::isUseExponentialBackOff ( ) const [inline, virtual]`

#### Returns

whether or not the exponential back off option is enabled.

Implements **activemq::core::RedeliveryPolicy** (p. 3271).

**6.307.2.9** `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setBackOffMultiplier ( double value ) [inline, virtual]`

Sets the Back-Off Multiplier for Message Redelivery.

#### Parameters

<i>value</i>	The new value for the back-off multiplier.
--------------	--

Implements **activemq::core::RedeliveryPolicy** (p. 3271).

**6.307.2.10** `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setCollisionAvoidancePercent ( short value ) [virtual]`

#### Parameters

<i>value</i>	The collision avoidance percentage setting.
--------------	---

Implements **activemq::core::RedeliveryPolicy** (p. 3271).

**6.307.2.11** `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setInitialRedeliveryDelay ( long long value ) [inline, virtual]`

Sets the initial time that redelivery will be delayed.

#### Parameters

<i>value</i>	Time in Milliseconds to wait before starting redelivery.
--------------	--

Implements **activemq::core::RedeliveryPolicy** (p. 3272).

**6.307.2.12** `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setMaximumRedeliveries ( int maximumRedeliveries ) [inline, virtual]`

Sets the Maximum allowable redeliveries for a Message.

#### Parameters

<i>maximum-Redeliveries</i>	The maximum number of times that a message will be redelivered.
-----------------------------	---

Implements `activemq::core::RedeliveryPolicy` (p. 3272).

**6.307.2.13** `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setUseCollisionAvoidance ( bool value ) [inline, virtual]`

#### Parameters

<i>value</i>	Enable or Disable collision avoidance for this Policy.
--------------	--

Implements `activemq::core::RedeliveryPolicy` (p. 3272).

**6.307.2.14** `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setUseExponentialBackOff ( bool value ) [inline, virtual]`

#### Parameters

<i>value</i>	Enable or Disable the exponential back off multiplier option.
--------------	---

Implements `activemq::core::RedeliveryPolicy` (p. 3272).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/policies/DefaultRedeliveryPolicy.h`

## 6.308 decaf::internal::net::DefaultServerSocketFactory Class Reference

Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.

```
#include <src/main/decaf/internal/net/DefaultServerSocketFactory.h>
```

Inheritance diagram for `decaf::internal::net::DefaultServerSocketFactory`:



## Public Member Functions

- **DefaultServerSocketFactory** ()
- virtual **~DefaultServerSocketFactory** ()
- virtual **decaf::net::ServerSocket \* createServerSocket** ()

Create a new *ServerSocket* (p. 3447) that is unbound.  
The *ServerSocket* (p. 3447) will have been configured with the defaults from the factory.

**Returns**

new *ServerSocket* (p. 3447) pointer that is owned by the caller.

**Exceptions**

IOException	if the <i>ServerSocket</i> (p. 3447) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket \* createServerSocket** (int port)

Create a new *ServerSocket* (p. 3447) that is bound to the given port.  
The *ServerSocket* (p. 3447) will have been configured with the defaults from the factory.

**Parameters**

port	The port to bind the <i>ServerSocket</i> (p. 3447) to.
------	--

**Returns**

new *ServerSocket* (p. 3447) pointer that is owned by the caller.

**Exceptions**

IOException	if the <i>ServerSocket</i> (p. 3447) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket \* createServerSocket** (int port, int backlog)

Create a new *ServerSocket* (p. 3447) that is bound to the given port.  
The *ServerSocket* (p. 3447) will have been configured with the defaults from the factory. The *ServerSocket* (p. 3447) will use the specified connection backlog setting.

**Parameters**

port	The port to bind the <i>ServerSocket</i> (p. 3447) to.
backlog	The number of pending connect request the <i>ServerSocket</i> (p. 3447) can queue.

**Returns**

new *ServerSocket* (p. 3447) pointer that is owned by the caller.

**Exceptions**

IOException	if the <i>ServerSocket</i> (p. 3447) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket \* createServerSocket** (int port, int backlog, const **decaf::net::InetAddress** \*address)

Create a new *ServerSocket* (p. 3447) that is bound to the given port.  
The *ServerSocket* (p. 3447) will have been configured with the defaults from the factory. The *ServerSocket* (p. 3447) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the *ServerSocket* (p. 3447) will listen on all interfaces.

**Parameters**

port	<i>The port to bind the <b>ServerSocket</b> (p. 3447) to.</i>
backlog	<i>The number of pending connect request the <b>ServerSocket</b> (p. 3447) can queue.</i>
address	<i>The address of the interface on the local machine to bind to.</i>

**Returns**

*new **ServerSocket** (p. 3447) pointer that is owned by the caller.*

**Exceptions**

IOException	<i>if the <b>ServerSocket</b> (p. 3447) cannot be created for some reason.</i>
-------------	--

**6.308.1 Detailed Description**

Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.

**Since**

1.0

**6.308.2 Constructor & Destructor Documentation**

**6.308.2.1** `decaf::internal::net::DefaultServerSocketFactory::DefaultServerSocketFactory ( )`

**6.308.2.2** `virtual decaf::internal::net::DefaultServerSocketFactory::~~DefaultServerSocketFactory ( ) [virtual]`

**6.308.3 Member Function Documentation**

**6.308.3.1** `virtual decaf::net::ServerSocket* decaf::internal::net::DefaultServerSocketFactory::createServerSocket ( ) [virtual]`

Create a new **ServerSocket** (p. 3447) that is unbound.

The **ServerSocket** (p. 3447) will have been configured with the defaults from the factory.

**Returns**

*new **ServerSocket** (p. 3447) pointer that is owned by the caller.*

**Exceptions**

IOException	<i>if the <b>ServerSocket</b> (p. 3447) cannot be created for some reason.</i>
-------------	--

Reimplemented from **decaf::net::ServerSocketFactory** (p. 3458).

**6.308.3.2** virtual decaf::net::ServerSocket\* decaf::internal::net::DefaultServerSocketFactory::createServerSocket ( int *port*, int *backlog*, const decaf::net::InetAddress \* *address* ) [virtual]

Create a new **ServerSocket** (p. 3447) that is bound to the given port.

The **ServerSocket** (p. 3447) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3447) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 3447) will listen on all interfaces.

#### Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3447) to.
<i>backlog</i>	The number of pending connect request the <b>ServerSocket</b> (p. 3447) can queue.
<i>address</i>	The address of the interface on the local machine to bind to.

#### Returns

new **ServerSocket** (p. 3447) pointer that is owned by the caller.

#### Exceptions

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3447) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 3459).

**6.308.3.3** virtual decaf::net::ServerSocket\* decaf::internal::net::DefaultServerSocketFactory::createServerSocket ( int *port*, int *backlog* ) [virtual]

Create a new **ServerSocket** (p. 3447) that is bound to the given port.

The **ServerSocket** (p. 3447) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3447) will use the specified connection backlog setting.

#### Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3447) to.
<i>backlog</i>	The number of pending connect request the <b>ServerSocket</b> (p. 3447) can queue.

#### Returns

new **ServerSocket** (p. 3447) pointer that is owned by the caller.

#### Exceptions

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3447) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 3459).

**6.308.3.4** `virtual decaf::net::ServerSocket* decaf::internal::net::DefaultServerSocketFactory::createServerSocket ( int port )`  
`[virtual]`

Create a new **ServerSocket** (p. 3447) that is bound to the given port.

The **ServerSocket** (p. 3447) will have been configured with the defaults from the factory.

#### Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3447) to.
-------------	--

#### Returns

new **ServerSocket** (p. 3447) pointer that is owned by the caller.

#### Exceptions

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3447) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 3458).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/DefaultServerSocketFactory.h`

## 6.309 decaf::internal::net::DefaultSocketFactory Class Reference

SocketFactory implementation that is used to create Sockets.

```
#include <src/main/decaf/internal/net/DefaultSocketFactory.h>
```

Inheritance diagram for `decaf::internal::net::DefaultSocketFactory`:

#### Public Member Functions

- **DefaultSocketFactory** ()
- virtual **~DefaultSocketFactory** ()
- virtual **decaf::net::Socket \* createSocket** () throw ( decaf::io::IOException )

*Creates an unconnected **Socket** (p. 3607) object.*

#### Returns

*a new **Socket** (p. 3607) object, caller must free this object when done.*

#### Exceptions

<i>IOException</i>	if the <b>Socket</b> (p. 3607) cannot be created.
--------------------	---

- virtual **decaf::net::Socket \* createSocket** (const **decaf::net::InetAddress** \*host, int port) throw ( **decaf::io::IOException**, **decaf::net::UnknownHostException** )

*Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).*

**Parameters**

host	<i>The host to connect the socket to.</i>
port	<i>The port on the remote host to connect to.</i>

**Returns**

*a new **Socket** (p. 3607) object, caller must free this object when done.*

**Exceptions**

<b>IOException</b>	<i>if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.</i>
<b>UnknownHostException</b> (p. 4019)	<i>if the host name is not known.</i>

- virtual **decaf::net::Socket \* createSocket** (const **decaf::net::InetAddress** \*host, int port, const **decaf::net::InetAddress** \*ifAddress, int localPort) throw ( **decaf::io::IOException**, **decaf::net::UnknownHostException** )

*Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).*

*The **Socket** (p. 3607) will be bound to the specified local address and port.*

**Parameters**

host	<i>The host to connect the socket to.</i>
port	<i>The port on the remote host to connect to.</i>
ifAddress	<i>The address on the local machine to bind the <b>Socket</b> (p. 3607) to.</i>
localPort	<i>The local port to bind the <b>Socket</b> (p. 3607) to.</i>

**Returns**

*a new **Socket** (p. 3607) object, caller must free this object when done.*

**Exceptions**

<b>IOException</b>	<i>if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.</i>
<b>UnknownHostException</b> (p. 4019)	<i>if the host name is not known.</i>

- virtual **decaf::net::Socket \* createSocket** (const std::string &name, int port) throw ( **decaf::io::IOException**, **decaf::net::UnknownHostException** )

*Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).*

**Parameters**

host	<i>The host name or IP address to connect the socket to.</i>
port	<i>The port on the remote host to connect to.</i>

**Returns**

*a new **Socket** (p. 3607) object, caller must free this object when done.*

*Exceptions*

IOException	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<b>UnknownHostException</b> (p. 4019)	if the host name is not known.

- virtual **decaf::net::Socket \* createSocket** (const std::string &name, int port, const **decaf::net::InetAddress** \*ifAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

*Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).*

*Parameters*

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.
ifAddress	The address on the local machine to bind the <b>Socket</b> (p. 3607) to.
localPort	The local port to bind the <b>Socket</b> (p. 3607) to.

*Returns*

*a new **Socket** (p. 3607) object, caller must free this object when done.*

*Exceptions*

IOException	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<b>UnknownHostException</b> (p. 4019)	if the host name is not known.

**6.309.1 Detailed Description**

SocketFactory implementation that is used to create Sockets.

Since

1.0

## 6.309.2 Constructor & Destructor Documentation

6.309.2.1 decaf::internal::net::DefaultSocketFactory::DefaultSocketFactory ( )

6.309.2.2 virtual decaf::internal::net::DefaultSocketFactory::~DefaultSocketFactory ( )  
[virtual]

## 6.309.3 Member Function Documentation

6.309.3.1 virtual decaf::net::Socket\* decaf::internal::net::DefaultSocketFactory::createSocket  
( ) throw ( decaf::io::IOException ) [virtual]

Creates an unconnected **Socket** (p. 3607) object.

### Returns

a new **Socket** (p. 3607) object, caller must free this object when done.

### Exceptions

<i>IOException</i>	if the <b>Socket</b> (p. 3607) cannot be created.
--------------------	---

Reimplemented from **decaf::net::SocketFactory** (p. 3631).

6.309.3.2 virtual decaf::net::Socket\* decaf::internal::net::DefaultSocketFactory::createSocket  
( const std::string & *name*, int *port*, const decaf::net::InetAddress  
\* *ifAddress*, int *localPort* ) throw ( decaf::io::IOException,  
decaf::net::UnknownHostException ) [virtual]

Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).

### Parameters

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the <b>Socket</b> (p. 3607) to.
<i>localPort</i>	The local port to bind the <b>Socket</b> (p. 3607) to.

### Returns

a new **Socket** (p. 3607) object, caller must free this object when done.

### Exceptions

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<i>UnknownHostException</i> (p. 4019)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3632).

**6.309.3.3** `virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket  
( const std::string & name, int port ) throw ( decaf::io::IOException,  
decaf::net::UnknownHostException ) [virtual]`

Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).

#### Parameters

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

#### Returns

a new **Socket** (p. 3607) object, caller must free this object when done.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<i>UnknownHostException</i> (p. 4019)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3632).

**6.309.3.4** `virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket  
( const decaf::net::InetAddress * host, int port, const de-  
caf::net::InetAddress * ifAddress, int localPort ) throw ( decaf::io::IOException, decaf::net::UnknownHostException )  
[virtual]`

Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).

The **Socket** (p. 3607) will be bound to the specified local address and port.

#### Parameters

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the <b>Socket</b> (p. 3607) to.
<i>localPort</i>	The local port to bind the <b>Socket</b> (p. 3607) to.

#### Returns

a new **Socket** (p. 3607) object, caller must free this object when done.



**Exceptions**

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<b>UnknownHostException</b> (p. 4019)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3633).

**6.309.3.5** `virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket  
( const decaf::net::InetAddress * host, int port ) throw ( de-  
caf::io::IOException, decaf::net::UnknownHostException )  
[virtual]`

Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).

**Parameters**

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

**Returns**

a new **Socket** (p. 3607) object, caller must free this object when done.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<b>UnknownHostException</b> (p. 4019)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3631).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**DefaultSocketFactory.h**

**6.310 decaf::internal::net::ssl::DefaultSSLContext Class Reference**

Default SSLContext manager for the Decaf library.

```
#include <src/main/decaf/internal/net/ssl/DefaultSSLContext.h>
```

**Public Member Functions**

- virtual **~DefaultSSLContext** ()

## Static Public Member Functions

- static `decaf::net::ssl::SSLContext * getContext ()`

## Protected Member Functions

- `DefaultSSLContext ()`

### 6.310.1 Detailed Description

Default SSLContext manager for the Decaf library. If the user doesn't supply or specify the SSLContext that they wish to use then we load the Decaf library's default SSLContext using whatever SSL provider is enabled an preferred.

Since

1.0

### 6.310.2 Constructor & Destructor Documentation

6.310.2.1 `decaf::internal::net::ssl::DefaultSSLContext::DefaultSSLContext ( )`  
[protected]

6.310.2.2 `virtual decaf::internal::net::ssl::DefaultSSLContext::~~DefaultSSLContext ( )`  
[virtual]

### 6.310.3 Member Function Documentation

6.310.3.1 `static decaf::net::ssl::SSLContext* decaf::internal::net::ssl::DefaultSSLContext::getContext ( )`  
[static]

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/DefaultSSLContext.h`

## 6.311 decaf::internal::net::ssl::DefaultSSLServerSocketFactory Class Reference

Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

```
#include <src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h>
```

## 6.311 decaf::internal::net::ssl::DefaultSSLServerSocketFactory Class Reference 1747

Inheritance diagram for decaf::internal::net::ssl::DefaultSSLServerSocketFactory:

### Public Member Functions

- **DefaultSSLServerSocketFactory** (const std::string &errorMessage)
- virtual ~**DefaultSSLServerSocketFactory** ()
- virtual **decaf::net::ServerSocket \* createServerSocket** ()

Create a new *ServerSocket* (p. 3447) that is unbound.

The *ServerSocket* (p. 3447) will have been configured with the defaults from the factory.

**Returns**

new *ServerSocket* (p. 3447) pointer that is owned by the caller.

**Exceptions**

IOException	if the <i>ServerSocket</i> (p. 3447) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket \* createServerSocket** (int port)

Create a new *ServerSocket* (p. 3447) that is bound to the given port.

The *ServerSocket* (p. 3447) will have been configured with the defaults from the factory.

**Parameters**

port	The port to bind the <i>ServerSocket</i> (p. 3447) to.
------	--

**Returns**

new *ServerSocket* (p. 3447) pointer that is owned by the caller.

**Exceptions**

IOException	if the <i>ServerSocket</i> (p. 3447) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket \* createServerSocket** (int port, int backlog)

Create a new *ServerSocket* (p. 3447) that is bound to the given port.

The *ServerSocket* (p. 3447) will have been configured with the defaults from the factory. The *ServerSocket* (p. 3447) will use the specified connection backlog setting.

**Parameters**

port	The port to bind the <i>ServerSocket</i> (p. 3447) to.
backlog	The number of pending connect request the <i>ServerSocket</i> (p. 3447) can queue.

**Returns**

new *ServerSocket* (p. 3447) pointer that is owned by the caller.

**Exceptions**

IOException	if the <i>ServerSocket</i> (p. 3447) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket \* createServerSocket** (int port, int backlog,

const **decaf::net::InetAddress** \*address)

Create a new **ServerSocket** (p. 3447) that is bound to the given port. The **ServerSocket** (p. 3447) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3447) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 3447) will listen on all interfaces.

**Parameters**

port	The port to bind the <b>ServerSocket</b> (p. 3447) to.
backlog	The number of pending connect request the <b>ServerSocket</b> (p. 3447) can queue.
address	The address of the interface on the local machine to bind to.

**Returns**

new **ServerSocket** (p. 3447) pointer that is owned by the caller.

**Exceptions**

IOException	if the <b>ServerSocket</b> (p. 3447) cannot be created for some reason.
-------------	---

- virtual std::vector< std::string > **getDefaultCipherSuites** ()

Returns the list of cipher suites which are enabled by default. Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

**Returns**

an STL vector containing the list of cipher suites enabled by default.

**See also**

**getSupportedCipherSuites**() (p. 3670)

- virtual std::vector< std::string > **getSupportedCipherSuites** ()

Returns the names of the cipher suites which could be enabled for use on an SSL connection. Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

**Returns**

an STL vector containing the list of supported cipher suites.

**See also**

**getDefaultCipherSuites**() (p. 3669)

### 6.311.1 Detailed Description

Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

**Since**

1.0

## 6.311 `decaf::internal::net::ssl::DefaultSSLServerSocketFactory` Class Reference 1749

### 6.311.2 Constructor & Destructor Documentation

6.311.2.1 `decaf::internal::net::ssl::DefaultSSLServerSocketFactory::DefaultSSLServerSocketFactory ( const std::string & errorMessage )`

6.311.2.2 `virtual decaf::internal::net::ssl::DefaultSSLServerSocketFactory::~DefaultSSLServerSocketFactory ( ) [virtual]`

### 6.311.3 Member Function Documentation

6.311.3.1 `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket ( ) [virtual]`

Create a new **ServerSocket** (p. 3447) that is unbound.

The **ServerSocket** (p. 3447) will have been configured with the defaults from the factory.

#### Returns

new **ServerSocket** (p. 3447) pointer that is owned by the caller.

#### Exceptions

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3447) cannot be created for some reason.
--------------------	---

Reimplemented from **decaf::net::ServerSocketFactory** (p. 3458).

6.311.3.2 `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket ( int port ) [virtual]`

Create a new **ServerSocket** (p. 3447) that is bound to the given port.

The **ServerSocket** (p. 3447) will have been configured with the defaults from the factory.

#### Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3447) to.
-------------	--

#### Returns

new **ServerSocket** (p. 3447) pointer that is owned by the caller.

#### Exceptions

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3447) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 3458).

**6.311.3.3** `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket ( int port, int backlog, const decaf::net::InetAddress * address ) [virtual]`

Create a new **ServerSocket** (p. 3447) that is bound to the given port.

The **ServerSocket** (p. 3447) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3447) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 3447) will listen on all interfaces.

#### Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3447) to.
<i>backlog</i>	The number of pending connect request the <b>ServerSocket</b> (p. 3447) can queue.
<i>address</i>	The address of the interface on the local machine to bind to.

#### Returns

new **ServerSocket** (p. 3447) pointer that is owned by the caller.

#### Exceptions

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3447) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 3459).

**6.311.3.4** `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket ( int port, int backlog ) [virtual]`

Create a new **ServerSocket** (p. 3447) that is bound to the given port.

The **ServerSocket** (p. 3447) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3447) will use the specified connection backlog setting.

#### Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3447) to.
<i>backlog</i>	The number of pending connect request the <b>ServerSocket</b> (p. 3447) can queue.

#### Returns

new **ServerSocket** (p. 3447) pointer that is owned by the caller.

#### Exceptions

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3447) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 3459).

## 6.312 `decaf::internal::net::ssl::DefaultSSLSocketFactory` Class Reference 1751

**6.311.3.5** `virtual std::vector<std::string> decaf::internal::net::ssl::DefaultSSLServerSocketFactory::getDefaultCipherSuites ( )`  
[virtual]

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

### Returns

an STL vector containing the list of cipher suites enabled by default.

### See also

`getSupportedCipherSuites()` (p. 3670)

Implements `decaf::net::ssl::SSLServerSocketFactory` (p. 3669).

**6.311.3.6** `virtual std::vector<std::string> decaf::internal::net::ssl::DefaultSSLServerSocketFactory::getSupportedCipherSuites ( )`  
[virtual]

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

### Returns

an STL vector containing the list of supported cipher suites.

### See also

`getDefaultCipherSuites()` (p. 3669)

Implements `decaf::net::ssl::SSLServerSocketFactory` (p. 3670).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h`

## 6.312 `decaf::internal::net::ssl::DefaultSSLSocketFactory` Class Reference

Default implementation of the `SSLSocketFactory`, this factory throws an `Exception` from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

```
#include <src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h>
```

Inheritance diagram for decaf::internal::net::ssl::DefaultSSLSocketFactory:

## Public Member Functions

- **DefaultSSLSocketFactory** (const std::string &errorMessage)
- virtual ~**DefaultSSLSocketFactory** ()
- virtual **decaf::net::Socket** \* **createSocket** () throw ( decaf::io::IOException )

*Creates an unconnected **Socket** (p. 3607) object.*

### Returns

*a new **Socket** (p. 3607) object, caller must free this object when done.*

### Exceptions

IOException	<i>if the <b>Socket</b> (p. 3607) cannot be created.</i>
-------------	--

- virtual **decaf::net::Socket** \* **createSocket** (const **decaf::net::InetAddress** \*host, int port) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

*Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).*

### Parameters

host	<i>The host to connect the socket to.</i>
port	<i>The port on the remote host to connect to.</i>

### Returns

*a new **Socket** (p. 3607) object, caller must free this object when done.*

### Exceptions

IOException	<i>if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.</i>
<b>UnknownHostException</b> (p. 4019)	<i>if the host name is not known.</i>

- virtual **decaf::net::Socket** \* **createSocket** (const **decaf::net::InetAddress** \*host, int port, const **decaf::net::InetAddress** \*ifAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

*Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).*

*The **Socket** (p. 3607) will be bound to the specified local address and port.*

### Parameters

host	<i>The host to connect the socket to.</i>
port	<i>The port on the remote host to connect to.</i>
ifAddress	<i>The address on the local machine to bind the <b>Socket</b> (p. 3607) to.</i>
localPort	<i>The local port to bind the <b>Socket</b> (p. 3607) to.</i>



**Returns**

a new **Socket** (p. 3607) object, caller must free this object when done.

**Exceptions**

IOException	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<b>UnknownHostException</b> (p. 4019)	if the host name is not known.

- virtual **decaf::net::Socket \* createSocket** (const std::string &name, int port) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).

**Parameters**

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.

**Returns**

a new **Socket** (p. 3607) object, caller must free this object when done.

**Exceptions**

IOException	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<b>UnknownHostException</b> (p. 4019)	if the host name is not known.

- virtual **decaf::net::Socket \* createSocket** (const std::string &name, int port, const **decaf::net::InetAddress** \*ifAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).

**Parameters**

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.
ifAddress	The address on the local machine to bind the <b>Socket</b> (p. 3607) to.
localPort	The local port to bind the <b>Socket</b> (p. 3607) to.

**Returns**

a new **Socket** (p. 3607) object, caller must free this object when done.

**Exceptions**

IOException	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<b>UnknownHostException</b> (p. 4019)	if the host name is not known.

- virtual std::vector< std::string > **getDefaultCipherSuites** ()

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

#### Returns

an STL vector containing the list of cipher suites enabled by default.

#### See also

`getSupportedCipherSuites()` (p. 3682)

- virtual `std::vector< std::string > getSupportedCipherSuites ()`

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

#### Returns

an STL vector containing the list of supported cipher suites.

#### See also

`getDefaultCipherSuites()` (p. 3681)

- virtual `decaf::net::Socket * createSocket (decaf::net::Socket *socket, std::string host, int port, bool autoClose)`

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

#### Parameters

socket	The existing socket to layer over.
host	The server host the original <b>Socket</b> (p. 3607) is connected to.
port	The server port the original <b>Socket</b> (p. 3607) is connected to.
autoClose	Should the layered over <b>Socket</b> (p. 3607) be closed when the topmost socket is closed.

#### Returns

a new **Socket** (p. 3607) instance that wraps the given **Socket** (p. 3607).

#### Exceptions

<b>IOException</b>	if an I/O exception occurs while performing this operation.
<b>UnknownHostException</b> (p. 4019)	if the host is unknown.

### 6.312.1 Detailed Description

Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

#### Since

1.0

### 6.312.2 Constructor & Destructor Documentation

**6.312.2.1** `decaf::internal::net::ssl::DefaultSSLSocketFactory::DefaultSSLSocketFactory ( const std::string & errorMessage )`

**6.312.2.2** `virtual decaf::internal::net::ssl::DefaultSSLSocketFactory::~~DefaultSSLSocketFactory ( ) [virtual]`

### 6.312.3 Member Function Documentation

**6.312.3.1** `virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket ( ) throw ( decaf::io::IOException ) [virtual]`

Creates an unconnected **Socket** (p. 3607) object.

#### Returns

a new **Socket** (p. 3607) object, caller must free this object when done.

#### Exceptions

<i>IOException</i>	if the <b>Socket</b> (p. 3607) cannot be created.
--------------------	---

Reimplemented from **decaf::net::SocketFactory** (p. 3631).

**6.312.3.2** `virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket ( decaf::net::Socket * socket, std::string host, int port, bool autoClose ) [virtual]`

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

#### Parameters

<i>socket</i>	The existing socket to layer over.
---------------	------------------------------------

<i>host</i>	The server host the original <b>Socket</b> (p. 3607) is connected to.
<i>port</i>	The server port the original <b>Socket</b> (p. 3607) is connected to.
<i>autoClose</i>	Should the layered over <b>Socket</b> (p. 3607) be closed when the topmost socket is closed.

**Returns**

a new **Socket** (p. 3607) instance that wraps the given **Socket** (p. 3607).

**Exceptions**

<i>IOException</i>	if an I/O exception occurs while performing this operation.
<b><i>UnknownHostException</i></b> (p. 4019)	if the host is unknown.

Implements **decaf::net::ssl::SSLSocketFactory** (p. 3680).

**6.312.3.3** `virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket ( const decaf::net::InetAddress * host, int port ) throw ( decaf::io::IOException, decaf::net::UnknownHostException )`  
[virtual]

Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).

**Parameters**

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

**Returns**

a new **Socket** (p. 3607) object, caller must free this object when done.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<b><i>UnknownHostException</i></b> (p. 4019)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3631).

## 6.312 decaf::internal::net::ssl::DefaultSSLSocketFactory Class Reference 1757

**6.312.3.4** virtual decaf::net::Socket\* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket ( const std::string & *name*, int *port* ) throw ( decaf::io::IOException, decaf::net::UnknownHostException ) [virtual]

Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).

### Parameters

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

### Returns

a new **Socket** (p. 3607) object, caller must free this object when done.

### Exceptions

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<b>UnknownHostException</b> (p. 4019)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3632).

**6.312.3.5** virtual decaf::net::Socket\* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket ( const std::string & *name*, int *port*, const decaf::net::InetAddress \* *ifAddress*, int *localPort* ) throw ( decaf::io::IOException, decaf::net::UnknownHostException ) [virtual]

Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).

### Parameters

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the <b>Socket</b> (p. 3607) to.
<i>localPort</i>	The local port to bind the <b>Socket</b> (p. 3607) to.

### Returns

a new **Socket** (p. 3607) object, caller must free this object when done.

### Exceptions

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<b>UnknownHostException</b> (p. 4019)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3632).

**6.312.3.6** `virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket ( const decaf::net::InetAddress * host, int port, const decaf::net::InetAddress * ifAddress, int localPort ) throw ( decaf::io::IOException, decaf::net::UnknownHostException )`  
[virtual]

Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).

The **Socket** (p. 3607) will be bound to the specified local address and port.

#### Parameters

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the <b>Socket</b> (p. 3607) to.
<i>localPort</i>	The local port to bind the <b>Socket</b> (p. 3607) to.

#### Returns

a new **Socket** (p. 3607) object, caller must free this object when done.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<i>UnknownHostException</i> (p. 4019)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3633).

**6.312.3.7** `virtual std::vector<std::string> decaf::internal::net::ssl::DefaultSSLSocketFactory::getDefaultCipherSuites ( )`  
[virtual]

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

#### Returns

an STL vector containing the list of cipher suites enabled by default.

#### See also

**getSupportedCipherSuites()** (p. 3682)

Implements **decaf::net::ssl::SSLSocketFactory** (p. 3681).

**6.312.3.8** `virtual std::vector<std::string> decaf::internal::net::ssl::DefaultSSLSocketFactory::getSupportedCipherSuites ( )`  
`[virtual]`

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

#### Returns

an STL vector containing the list of supported cipher suites.

#### See also

`getDefaultCipherSuites()` (p. 3681)

Implements `decaf::net::ssl::SSLSocketFactory` (p. 3682).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h`

## 6.313 activemq::transport::DefaultTransportListener Class Reference

```
#include <src/main/activemq/transport/DefaultTransportListener.h>
```

Inheritance diagram for `activemq::transport::DefaultTransportListener`:

### Public Member Functions

- `virtual ~DefaultTransportListener ()`
- `virtual void onCommand (const Pointer< Command > &command AMQCPP_UNUSED)`  
*Event handler for the receipt of a command.*
- `virtual void onException (const decaf::lang::Exception &ex AMQCPP_UNUSED)`  
*Event handler for an exception from a command transport.*
- `virtual void transportInterrupted ()`  
*The transport has suffered an interruption from which it hopes to recover.*
- `virtual void transportResumed ()`  
*The transport has resumed after an interruption.*

### 6.313.1 Constructor & Destructor Documentation

6.313.1.1 `virtual activemq::transport::DefaultTransportListener::~~DefaultTransportListener ( )`  
`[inline, virtual]`

### 6.313.2 Member Function Documentation

6.313.2.1 `virtual void activemq::transport::DefaultTransportListener::onCommand ( const`  
`Pointer< Command > &command AMQCPP_UNUSED ) [inline,`  
`virtual]`

Event handler for the receipt of a command.

The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3996) deletes the command upon receipt.

#### Parameters

<i>command</i>	the received command object.
----------------	------------------------------

6.313.2.2 `virtual void activemq::transport::DefaultTransportListener::onException ( const`  
`decaf::lang::Exception &ex AMQCPP_UNUSED ) [inline, virtual]`

Event handler for an exception from a command transport.

#### Parameters

<i>ex</i>	The exception.
-----------	----------------

6.313.2.3 `virtual void activemq::transport::DefaultTransportListener::transportInterrupted ( )`  
`[inline, virtual]`

The transport has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 4015).

6.313.2.4 `virtual void activemq::transport::DefaultTransportListener::transportResumed ( )`  
`[inline, virtual]`

The transport has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 4015).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/DefaultTransportListener.h`



## 6.314 decaf::util::zip::Deflater Class Reference

This class compresses data using the *DEFLATE* algorithm (see *specification*).

```
#include <src/main/decaf/util/zip/Deflater.h>
```

### Public Member Functions

- **Deflater** (int level, bool nowrap=false)  
*Creates a new compressor using the specified compression level.*
- **Deflater** ()  
*Creates a new compressor with the default compression level.*
- virtual **~Deflater** ()
- void **setInput** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )  
*Sets input data for compression.*
- void **setInput** (const std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )  
*Sets input data for compression.*
- void **setInput** (const std::vector< unsigned char > &buffer) throw ( decaf::lang::exceptions::IllegalStateException )  
*Sets input data for compression.*
- void **setDictionary** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )  
*Sets preset dictionary for compression.*
- void **setDictionary** (const std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )  
*Sets preset dictionary for compression.*
- void **setDictionary** (const std::vector< unsigned char > &buffer) throw ( decaf::lang::exceptions::IllegalStateException )  
*Sets preset dictionary for compression.*
- void **setStrategy** (int strategy) throw ( decaf::lang::exceptions::IllegalArgumentOutOfRangeException, decaf::lang::exceptions::IllegalStateException )  
*Sets the compression strategy to the specified value.*

- void **setLevel** (int level) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Sets the compression level to the specified value.*
- bool **needsInput** () const
- void **finish** ()  
*When called, indicates that compression should end with the current contents of the input buffer.*
- bool **finished** () const
- int **deflate** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )  
*Fills specified buffer with compressed data.*
- int **deflate** (std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )  
*Fills specified buffer with compressed data.*
- int **deflate** (std::vector< unsigned char > &buffer) throw ( decaf::lang::exceptions::IllegalStateException )  
*Fills specified buffer with compressed data.*
- long long **getAdler** () const throw ( decaf::lang::exceptions::IllegalStateException )
- long long **getBytesRead** () const throw ( decaf::lang::exceptions::IllegalStateException )
- long long **getBytesWritten** () const throw ( decaf::lang::exceptions::IllegalStateException )
- void **reset** () throw ( decaf::lang::exceptions::IllegalStateException )  
*Resets deflater so that a new set of input data can be processed.*
- void **end** ()  
*Closes the compressor and discards any unprocessed input.*

### Static Public Attributes

- static const int **BEST\_SPEED**  
*Compression level for fastest compression.*
- static const int **BEST\_COMPRESSION**  
*Compression level for best compression.*
- static const int **DEFAULT\_COMPRESSION**  
*Default compression level.*

- static const int **DEFLATED**  
*Compression method for the deflate algorithm (the only one currently supported).*
- static const int **NO\_COMPRESSION**  
*Compression level for no compression.*
- static const int **FILTERED**  
*Compression strategy best used for data consisting mostly of small values with a somewhat random distribution.*
- static const int **HUFFMAN\_ONLY**  
*Compression strategy for Huffman coding only.*
- static const int **DEFAULT\_STRATEGY**  
*Default compression strategy.*

### 6.314.1 Detailed Description

This class compresses data using the *DEFLATE* algorithm (see [specification](#)). Basically this class is part of the API to the stream based ZLIB compression library and is used as such by **DeflaterOutputStream** (p. 1769) and its descendants.

The typical usage of a **Deflater** (p. 1759) instance outside this package consists of a specific call to one of its constructors before being passed to an instance of **DeflaterOutputStream** (p. 1769).

See also

**DeflaterOutputStream** (p. 1769)

**Inflater** (p. 2088)

Since

1.0

### 6.314.2 Constructor & Destructor Documentation

#### 6.314.2.1 decaf::util::zip::Deflater::Deflater ( int *level*, bool *nowrap* = false )

Creates a new compressor using the specified compression level.

If 'nowrap' is true then the ZLIB header and checksum fields will not be used in order to support the compression format used in both GZIP and PKZIP.

#### Parameters

<i>level</i>	The compression level to use (0-9).
<i>nowrap</i>	If true uses GZip compatible compression (defaults to false).

### 6.314.2.2 decaf::util::zip::Deflater::Deflater ( )

Creates a new compressor with the default compression level.

Compressed data will be generated in ZLIB format.

### 6.314.2.3 virtual decaf::util::zip::Deflater::~~Deflater ( ) [virtual]

## 6.314.3 Member Function Documentation

### 6.314.3.1 int decaf::util::zip::Deflater::deflate ( unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

Fills specified buffer with compressed data.

Returns actual number of bytes of compressed data. A return value of 0 indicates that **needsInput()** (p. 1764) should be called in order to determine if more input data is required.

#### Parameters

<i>buffer</i>	The Buffer to write the compressed data to.
<i>size</i>	The size of the passed buffer.
<i>offset</i>	The position in the Buffer to start writing at.
<i>length</i>	The maximum number of byte of data to write.

#### Returns

the actual number of bytes of compressed data.

#### Exceptions

<i>NullPointerException</i>	if buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

### 6.314.3.2 int decaf::util::zip::Deflater::deflate ( std::vector< unsigned char > & *buffer*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

Fills specified buffer with compressed data.

Returns actual number of bytes of compressed data. A return value of 0 indicates that **needsInput()** (p. 1764) should be called in order to determine if more input data is required.

**Parameters**

<i>buffer</i>	The Buffer to write the compressed data to.
<i>offset</i>	The position in the Buffer to start writing at.
<i>length</i>	The maximum number of byte of data to write.

**Returns**

the actual number of bytes of compressed data.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

**6.314.3.3** `int decaf::util::zip::Deflater::deflate ( std::vector< unsigned char > & buffer ) throw ( decaf::lang::exceptions::IllegalStateException )`

Fills specified buffer with compressed data.

Returns actual number of bytes of compressed data. A return value of 0 indicates that **needsInput()** (p. 1764) should be called in order to determine if more input data is required.

**Parameters**

<i>buffer</i>	The Buffer to write the compressed data to.
---------------	---

**Returns**

the actual number of bytes of compressed data.

**Exceptions**

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

**6.314.3.4** `void decaf::util::zip::Deflater::end ( )`

Closes the compressor and discards any unprocessed input.

This method should be called when the compressor is no longer being used, but will also be called automatically by the destructor. Once this method is called, the behavior of the **Deflater** (p. 1759) object is undefined.

**6.314.3.5** `void decaf::util::zip::Deflater::finish ( )`

When called, indicates that compression should end with the current contents of the input buffer.

**6.314.3.6** `bool decaf::util::zip::Deflater::finished ( ) const`

#### Returns

true if the end of the compressed data output stream has been reached.

**6.314.3.7** `long long decaf::util::zip::Deflater::getAdler ( ) const throw ( decaf::lang::exceptions::IllegalStateException )`

#### Returns

the ADLER-32 value of the uncompressed data.

#### Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

**6.314.3.8** `long long decaf::util::zip::Deflater::getBytesRead ( ) const throw ( decaf::lang::exceptions::IllegalStateException )`

#### Returns

the total number of uncompressed bytes input so far.

#### Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

**6.314.3.9** `long long decaf::util::zip::Deflater::getBytesWritten ( ) const throw ( decaf::lang::exceptions::IllegalStateException )`

#### Returns

the total number of compressed bytes output so far.

#### Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

**6.314.3.10** `bool decaf::util::zip::Deflater::needsInput ( ) const`

#### Returns

true if the input data buffer is empty and **setInput()** (p. 1767) should be called in order to provide more input

**6.314.3.11** void decaf::util::zip::Deflater::reset ( ) throw ( decaf::lang::exceptions::IllegalStateException )

Resets deflater so that a new set of input data can be processed.

Keeps current compression level and strategy settings.

### Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

**6.314.3.12** void decaf::util::zip::Deflater::setDictionary ( const std::vector< unsigned char > & *buffer*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

Sets preset dictionary for compression.

A preset dictionary is used when the history buffer can be predetermined. When the data is later uncompressed with **Inflater.inflate()** (p. 2093), **Inflater.getAdler()** (p. 2091) can be called in order to get the Adler-32 value of the dictionary required for decompression.

### Parameters

<i>buffer</i>	The buffer containing the preset dictionary.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

### Exceptions

<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

**6.314.3.13** void decaf::util::zip::Deflater::setDictionary ( const unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

Sets preset dictionary for compression.

A preset dictionary is used when the history buffer can be predetermined. When the data is later uncompressed with **Inflater.inflate()** (p. 2093), **Inflater.getAdler()** (p. 2091) can be called in order to get the Adler-32 value of the dictionary required for decompression.

**Parameters**

<i>buffer</i>	The buffer containing the preset dictionary.
<i>size</i>	The size of the passed dictionary buffer.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

**Exceptions**

<i>NullPointerException</i>	if buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

**6.314.3.14** `void decaf::util::zip::Deflater::setDictionary ( const std::vector< unsigned char > & buffer ) throw ( decaf::lang::exceptions::IllegalStateException )`

Sets preset dictionary for compression.

A preset dictionary is used when the history buffer can be predetermined. When the data is later uncompressed with **Inflater.inflate()** (p. 2093), **Inflater.getAdler()** (p. 2091) can be called in order to get the Adler-32 value of the dictionary required for decompression.

**Parameters**

<i>buffer</i>	The buffer containing the preset dictionary.
---------------	--

**Exceptions**

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

**6.314.3.15** `void decaf::util::zip::Deflater::setInput ( const std::vector< unsigned char > & buffer ) throw ( decaf::lang::exceptions::IllegalStateException )`

Sets input data for compression.

This should be called whenever **needsInput()** (p. 1764) returns true indicating that more input data is required.

**Parameters**

<i>buffer</i>	The Buffer to read in for compression.
---------------	--

**Exceptions**

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------



**6.314.3.16** void decaf::util::zip::Deflater::setInput ( const std::vector< unsigned char > & *buffer*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

Sets input data for compression.

This should be called whenever **needsInput()** (p. 1764) returns true indicating that more input data is required.

#### Parameters

<i>buffer</i>	The Buffer to read in for compression.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

**6.314.3.17** void decaf::util::zip::Deflater::setInput ( const unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

Sets input data for compression.

This should be called whenever **needsInput()** (p. 1764) returns true indicating that more input data is required.

#### Parameters

<i>buffer</i>	The Buffer to read in for compression.
<i>size</i>	The size in bytes of the buffer passed.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

#### Exceptions

<i>NullPointerException</i>	if buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

**6.314.3.18** `void decaf::util::zip::Deflater::setLevel ( int level ) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )`

Sets the compression level to the specified value.

#### Parameters

<i>level</i>	The new Compression level to use.
--------------	-----------------------------------

#### Exceptions

<i>IllegalArgumentException</i>	if the level value is invalid.
<i>IllegalStateException</i>	if in the end state.

**6.314.3.19** `void decaf::util::zip::Deflater::setStrategy ( int strategy ) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )`

Sets the compression strategy to the specified value.

#### Parameters

<i>strategy</i>	The new Compression strategy to use.
-----------------	--------------------------------------

#### Exceptions

<i>IllegalArgumentException</i>	if the strategy value is invalid.
<i>IllegalStateException</i>	if in the end state.

### 6.314.4 Field Documentation

**6.314.4.1** `const int decaf::util::zip::Deflater::BEST_COMPRESSION` [static]

Compression level for best compression.

**6.314.4.2** `const int decaf::util::zip::Deflater::BEST_SPEED` [static]

Compression level for fastest compression.

**6.314.4.3** `const int decaf::util::zip::Deflater::DEFAULT_COMPRESSION`  
[static]

Default compression level.

**6.314.4.4** `const int decaf::util::zip::Deflater::DEFAULT_STRATEGY` [static]

Default compression strategy.

**6.314.4.5** `const int decaf::util::zip::Deflater::DEFLATED` [static]

Compression method for the deflate algorithm (the only one currently supported).

**6.314.4.6** `const int decaf::util::zip::Deflater::FILTERED` [static]

Compression strategy best used for data consisting mostly of small values with a somewhat random distribution.

Forces more Huffman coding and less string matching.

**6.314.4.7** `const int decaf::util::zip::Deflater::HUFFMAN_ONLY` [static]

Compression strategy for Huffman coding only.

**6.314.4.8** `const int decaf::util::zip::Deflater::NO_COMPRESSION` [static]

Compression level for no compression.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/Deflater.h`

**6.315 decaf::util::zip::DeflaterOutputStream Class Reference**

Provides a `FilterOutputStream` instance that compresses the data before writing it to the wrapped `OutputStream`.

```
#include <src/main/decaf/util/zip/DeflaterOutputStream.h>
```

Inheritance diagram for `decaf::util::zip::DeflaterOutputStream`:

**Public Member Functions**

- **DeflaterOutputStream** (`decaf::io::OutputStream *outputStream`, `bool own=false`)

*Creates a new `DeflateOutputStream` with a Default **Deflater** (p. 1759) and buffer size.*

- **DeflaterOutputStream** (`decaf::io::OutputStream *outputStream`, `Deflater *deflater`, `bool own=false`)

*Creates a new `DeflateOutputStream` with a user supplied **Deflater** (p. 1759) and a default buffer size.*

- **DeflaterOutputStream** (decaf::io::OutputStream \*outputStream, Deflater \*deflater, int bufferSize, bool own=false)

*Creates a new `DeflateOutputStream` with a user supplied **Deflater** (p. 1759) and specified buffer size.*

- virtual ~**DeflaterOutputStream** ()
- virtual void **finish** () throw ( decaf::io::IOException )

*Finishes writing any remaining data to the wrapped `OutputStream` but does not close it upon completion.*

- virtual void **close** () throw ( decaf::io::IOException )

*Closes this object and deallocates the appropriate resources.  
The object is generally no longer usable after calling close.*

#### **Exceptions**

<b>IOException</b> (p. 2210) if an error occurs while closing.
--

*The default implementation of this method does nothing.*

*The close method of **FilterOutputStream** (p. 1957) calls its flush method, and then calls the close method of its underlying output stream.*

## Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual void **deflate** () throw ( decaf::io::IOException )

*Writes a buffers worth of compressed data to the wrapped `OutputStream`.*

## Protected Attributes

- **Deflater \* deflater**

*The **Deflater** (p. 1759) for this stream.*

- std::vector< unsigned char > **buf**

*The Buffer to use for.*

- bool **ownDeflater**
- bool **isDone**

## Static Protected Attributes

- static const std::size\_t **DEFAULT\_BUFFER\_SIZE**

### 6.315.1 Detailed Description

Provides a FilterOutputStream instance that compresses the data before writing it to the wrapped OutputStream.

#### Since

1.0

### 6.315.2 Constructor & Destructor Documentation

**6.315.2.1** `decaf::util::zip::DeflaterOutputStream::DeflaterOutputStream (   
 decaf::io::OutputStream * outputStream, bool own = false )`

Creates a new DeflateOutputStream with a Default **Deflater** (p. 1759) and buffer size.

#### Parameters

<i>output-Stream</i>	The OutputStream instance to wrap.
<i>own</i>	Should this filter take ownership of the OutputStream pointer (default is false).

**6.315.2.2** `decaf::util::zip::DeflaterOutputStream::DeflaterOutputStream (   
 decaf::io::OutputStream * outputStream, Deflater * deflater, bool own =   
 false )`

Creates a new DeflateOutputStream with a user supplied **Deflater** (p. 1759) and a default buffer size.

When the user supplied a **Deflater** (p. 1759) instance the DeflaterOutpotStream does not take ownership of the **Deflater** (p. 1759) pointer, the caller is still responsible for deleting the **Deflater** (p. 1759).

#### Parameters

<i>output-Stream</i>	The OutputStream instance to wrap.
<i>deflater</i>	The user supplied <b>Deflater</b> (p. 1759) to use for compression. (
<i>own</i>	Should this filter take ownership of the OutputStream pointer (default is false).

## Exceptions

<i>NullPointerException</i>	if the <b>Deflater</b> (p. 1759) given is NULL.
-----------------------------	---

**6.315.2.3** `decaf::util::zip::DeflaterOutputStream::DeflaterOutputStream ( decaf::io::OutputStream * outputStream, Deflater * deflater, int bufferSize, bool own = false )`

Creates a new DeflateOutputStream with a user supplied **Deflater** (p. 1759) and specified buffer size.

When the user supplied a **Deflater** (p. 1759) instance the DeflaterOutputStream does not take ownership of the **Deflater** (p. 1759) pointer, the caller is still responsible for deleting the **Deflater** (p. 1759).

#### Parameters

<i>output-Stream</i>	The OutputStream instance to wrap.
<i>deflater</i>	The user supplied <b>Deflater</b> (p. 1759) to use for compression.
<i>bufferSize</i>	The size of the input buffer.
<i>own</i>	Should this filter take ownership of the OutputStream pointer (default is false).

#### Exceptions

<i>NullPointerException</i>	if the <b>Deflater</b> (p. 1759) given is NULL.
<i>IllegalArgumentException</i>	if bufferSize is 0.

**6.315.2.4** `virtual decaf::util::zip::DeflaterOutputStream::~~DeflaterOutputStream ( )`  
[virtual]

### 6.315.3 Member Function Documentation

**6.315.3.1** `virtual void decaf::util::zip::DeflaterOutputStream::close ( ) throw ( decaf::io::IOException )` [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

#### Exceptions

<i>IOException</i> (p. 2210)	if an error occurs while closing.
------------------------------	-----------------------------------

The default implementation of this method does nothing.

The close method of **FilterOutputStream** (p. 1957) calls its flush method, and then calls the close method of its underlying output stream.

Finishes writing any remaining data to the OutputStream then closes the stream.

Reimplemented from **decaf::io::FilterOutputStream** (p. 1959).

**6.315.3.2** `virtual void decaf::util::zip::DeflaterOutputStream::deflate ( ) throw ( decaf::io::IOException )` [protected, virtual]

Writes a buffers worth of compressed data to the wrapped OutputStream.

**6.315.3.3** `virtual void decaf::util::zip::DeflaterOutputStream::doWriteArrayBounded ( const unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1960).

**6.315.3.4** `virtual void decaf::util::zip::DeflaterOutputStream::doWriteByte ( unsigned char value ) throw ( decaf::io::IOException )` [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1960).

**6.315.3.5** `virtual void decaf::util::zip::DeflaterOutputStream::finish ( ) throw ( decaf::io::IOException )` [virtual]

Finishes writing any remaining data to the wrapped OutputStream but does not close it upon completion.

### Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

## 6.315.4 Field Documentation

**6.315.4.1** `std::vector<unsigned char> decaf::util::zip::DeflaterOutputStream::buf` [protected]

The Buffer to use for.

**6.315.4.2** `const std::size_t decaf::util::zip::DeflaterOutputStream::DEFAULT_BUFFER_SIZE` [static, protected]

**6.315.4.3** `Deflater* decaf::util::zip::DeflaterOutputStream::deflater` [protected]

The **Deflater** (p. 1759) for this stream.

**6.315.4.4** `bool decaf::util::zip::DeflaterOutputStream::isDone` [protected]

**6.315.4.5** `bool decaf::util::zip::DeflaterOutputStream::ownDeflater` [protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/DeflaterOutputStream.h`

## 6.316 decaf::util::concurrent::Delayed Class Reference

A mix-in style interface for marking objects that should be acted upon after a given delay.

```
#include <src/main/decaf/util/concurrent/Delayed.h>
```

Inheritance diagram for `decaf::util::concurrent::Delayed`:

### Public Member Functions

- virtual `~Delayed()`
- virtual long long `getDelay` (const **TimeUnit** &unit)=0

*Returns the remaining delay associated with this object, in the given time unit.*

#### 6.316.1 Detailed Description

A mix-in style interface for marking objects that should be acted upon after a given delay. An implementation of this interface must define a `Comparable` methods that provides an ordering consistent with its `getDelay` method.



### 6.316.2 Constructor & Destructor Documentation

6.316.2.1 virtual decaf::util::concurrent::Delayed::~Delayed ( ) [inline, virtual]

### 6.316.3 Member Function Documentation

6.316.3.1 virtual long long decaf::util::concurrent::Delayed::getDelay ( const TimeUnit & unit ) [pure virtual]

Returns the remaining delay associated with this object, in the given time unit.

#### Parameters

<i>unit</i>	The time unit
-------------	---------------

#### Returns

the remaining delay; zero or negative values indicate that the delay has already elapsed

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/Delayed.h

## 6.317 cms::DeliveryMode Class Reference

This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.

```
#include <src/main/cms/DeliveryMode.h>
```

### Public Types

- enum **DELIVERY\_MODE** { **PERSISTENT** = 0, **NON\_PERSISTENT** = 1 }

*Enumeration values for **Message** (p. 2614) Delivery Mode.*

### Public Member Functions

- virtual ~**DeliveryMode** ()

#### 6.317.1 Detailed Description

This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages. When a client sends a **cms::Message** (p. 2614)

it can mark the **Message** (p. 2614) as either Persistent or Non-Persistent. If the client feels that the **Message** (p. 2614) cannot be lost in transit it should mark it as Persistent, otherwise if it is allowable for a **Message** (p. 2614) to occasionally be lost it can mark it as Non-Persistent. This allows the Provider to balance make tradeoffs between balance and **Message** (p. 2614) throughput.

The **DeliveryMode** (p. 1775) covers only the transport of the **Message** (p. 2614) for sending client to its destination and doesn't apply to the receiving **Message** (p. 2614) consumer. The receiving Consumer can drop Message's based on configuration such as memory limits or **Message** (p. 2614) filtering.

A message is guaranteed to be delivered once and only once by a CMS provider if the delivery mode of the message is PERSISTENT and the configuration of the **Message** (p. 2614) consumer allows for it.

Since

1.0

### 6.317.2 Member Enumeration Documentation

#### 6.317.2.1 enum cms::DeliveryMode::DELIVERY\_MODE

Enumeration values for **Message** (p. 2614) Delivery Mode.

Enumerator:

*PERSISTENT*

*NON\_PERSISTENT*

### 6.317.3 Constructor & Destructor Documentation

#### 6.317.3.1 virtual cms::DeliveryMode::~DeliveryMode ( ) [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/cms/DeliveryMode.h

## 6.318 cms::Destination Class Reference

A **Destination** (p. 1776) object encapsulates a provider-specific address.

```
#include <src/main/cms/Destination.h>
```

Inheritance diagram for cms::Destination:

## Public Types

- enum **DestinationType** { **TOPIC**, **QUEUE**, **TEMPORARY\_TOPIC**, **TEMPORARY\_QUEUE** }

## Public Member Functions

- virtual **~Destination** ()
- virtual **DestinationType** **getDestinationType** () const =0  
*Retrieve the **Destination** (p. 1776) Type for this **Destination** (p. 1776).*
- virtual **cms::Destination** \* **clone** () const =0  
*Creates a new instance of this destination type that is a copy of this one, and returns it.*
- virtual void **copy** (const **cms::Destination** &source)=0  
*Copies the contents of the given **Destination** (p. 1776) object to this one.*
- virtual const **CMSProperties** & **getCMSProperties** () const =0  
*Retrieve any properties that might be part of the destination that was specified.*

### 6.318.1 Detailed Description

A **Destination** (p. 1776) object encapsulates a provider-specific address. There is no standard definition of a **Destination** (p. 1776) address, each provider can provide its own definition and there can be configuration data attached to the **Destination** (p. 1776) address.

All CMS **Destination** (p. 1776) objects support concurrent use.

#### Since

1.0

### 6.318.2 Member Enumeration Documentation

#### 6.318.2.1 enum cms::Destination::DestinationType

##### Enumerator:

**TOPIC**

**QUEUE**

**TEMPORARY\_TOPIC**

**TEMPORARY\_QUEUE**

### 6.318.3 Constructor & Destructor Documentation

6.318.3.1 `virtual cms::Destination::~~Destination ( ) [inline, virtual]`

### 6.318.4 Member Function Documentation

6.318.4.1 `virtual cms::Destination* cms::Destination::clone ( ) const [pure virtual]`

Creates a new instance of this destination type that is a copy of this one, and returns it.

#### Returns

cloned copy of this object

Implemented in `activemq::commands::ActiveMQQueue` (p. 480), `activemq::commands::ActiveMQTempQueue` (p. 608), `activemq::commands::ActiveMQTempTopic` (p. 638), and `activemq::commands::ActiveMQTopic` (p. 698).

6.318.4.2 `virtual void cms::Destination::copy ( const cms::Destination & source ) [pure virtual]`

Copies the contents of the given **Destination** (p. 1776) object to this one.

#### Parameters

<i>source</i>	The source <b>Destination</b> (p. 1776) object.
---------------	---

Implemented in `activemq::commands::ActiveMQQueue` (p. 481), `activemq::commands::ActiveMQTempQueue` (p. 608), `activemq::commands::ActiveMQTempTopic` (p. 638), and `activemq::commands::ActiveMQTopic` (p. 699).

6.318.4.3 `virtual const CMSProperties& cms::Destination::getCMSProperties ( ) const [pure virtual]`

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

#### Returns

A {const} reference to a **CMSProperties** (p. 1195) object.

Implemented in `activemq::commands::ActiveMQQueue` (p. 482), `activemq::commands::ActiveMQTempQueue` (p. 610), `activemq::commands::ActiveMQTempTopic` (p. 640), and `activemq::commands::ActiveMQTopic` (p. 700).

## 6.319 activemq::commands::ActiveMQDestination::DestinationFilter Struct Reference 1781

**6.318.4.4** `virtual DestinationType cms::Destination::getDestinationType ( ) const` [pure virtual]

Retrieve the **Destination** (p. 1776) Type for this **Destination** (p. 1776).

### Returns

The **Destination** (p. 1776) Type

Implemented in **activemq::commands::ActiveMQQueue** (p. 482), **activemq::commands::ActiveMQTempQueue** (p. 610), **activemq::commands::ActiveMQTempTopic** (p. 640), and **activemq::commands::ActiveMQTopic** (p. 700).

The documentation for this class was generated from the following file:

- `src/main/cms/Destination.h`

## 6.319 activemq::commands::ActiveMQDestination::DestinationFilter Struct Reference

```
#include <src/main/activemq/commands/ActiveMQDestination.h>
```

### Static Public Attributes

- static const std::string **ANY\_CHILD**
- static const std::string **ANY\_DESCENDENT**

### 6.319.1 Field Documentation

**6.319.1.1** `const std::string activemq::commands::ActiveMQDestination::DestinationFilter::ANY_CHILD` [static]

**6.319.1.2** `const std::string activemq::commands::ActiveMQDestination::DestinationFilter::ANY_DESCENDENT` [static]

The documentation for this struct was generated from the following file:

- `src/main/activemq/commands/ActiveMQDestination.h`

## 6.320 activemq::commands::DestinationInfo Class Reference

```
#include <src/main/activemq/commands/DestinationInfo.h>
```

Inheritance diagram for **activemq::commands::DestinationInfo**:

## Public Member Functions

- **DestinationInfo** ()
- virtual **~DestinationInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **DestinationInfo \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual unsigned char **getOperationType** () const
- virtual void **setOperationType** (unsigned char operationType)
- virtual long long **getTimeout** () const
- virtual void **setTimeout** (long long timeout)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &brokerPath)
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** \*visitor) throw ( **exceptions::ActiveMQException** )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

## Static Public Attributes

- static const unsigned char **ID\_DESTINATIONINFO** = 8

### Protected Attributes

- **Pointer< ConnectionId > connectionId**
- **Pointer< ActiveMQDestination > destination**
- unsigned char **operationType**
- long long **timeout**
- std::vector< **decaf::lang::Pointer< BrokerId > > brokerPath**

### 6.320.1 Constructor & Destructor Documentation

6.320.1.1 **activemq::commands::DestinationInfo::DestinationInfo ( )**

6.320.1.2 **virtual activemq::commands::DestinationInfo::~~DestinationInfo ( )** [virtual]

### 6.320.2 Member Function Documentation

6.320.2.1 **virtual DestinationInfo\* activemq::commands::DestinationInfo::cloneDataStructure ( ) const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

6.320.2.2 **virtual void activemq::commands::DestinationInfo::copyDataStructure ( const DataStructure \* src )** [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

6.320.2.3 **virtual bool activemq::commands::DestinationInfo::equals ( const DataStructure \* value ) const** [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

6.320.2.4 **virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::DestinationInfo::getBrokerPath ( ) const** [virtual]

6.320.2.5 **virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::DestinationInfo::getBrokerPath ( )** [virtual]

6.320.2.6 **virtual const Pointer<ConnectionId>& activemq::commands::DestinationInfo::getConnectionId ( ) const** [virtual]

6.320.2.7 **virtual Pointer<ConnectionId>& activemq::commands::DestinationInfo::getConnectionId ( )** [virtual]

6.320.2.8 **virtual unsigned char activemq::commands::DestinationInfo::getDataStructureType ( ) const** [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns**

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).



- 6.320.2.9 **virtual const Pointer<ActiveMQDestination>&activemq::commands::DestinationInfo::getDestination ( ) const** [virtual]
- 6.320.2.10 **virtual Pointer<ActiveMQDestination>&activemq::commands::DestinationInfo::getDestination ( )** [virtual]
- 6.320.2.11 **virtual unsigned char activemq::commands::DestinationInfo::getOperationType ( ) const** [virtual]
- 6.320.2.12 **virtual long long activemq::commands::DestinationInfo::getTimeout ( ) const** [virtual]
- 6.320.2.13 **virtual void activemq::commands::DestinationInfo::setBrokerPath ( const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath )** [virtual]
- 6.320.2.14 **virtual void activemq::commands::DestinationInfo::setConnectionId ( const Pointer< ConnectionId > & connectionId )** [virtual]
- 6.320.2.15 **virtual void activemq::commands::DestinationInfo::setDestination ( const Pointer< ActiveMQDestination > & destination )** [virtual]
- 6.320.2.16 **virtual void activemq::commands::DestinationInfo::setOperationType ( unsigned char operationType )** [virtual]
- 6.320.2.17 **virtual void activemq::commands::DestinationInfo::setTimeout ( long long timeout )** [virtual]
- 6.320.2.18 **virtual std::string activemq::commands::DestinationInfo::toString ( ) const** [virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

- 6.320.2.19 **virtual Pointer<Command> activemq::commands::DestinationInfo::visit ( activemq::state::CommandVisitor \* visitor ) throw ( exceptions::ActiveMQException )** [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

**Returns**

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

**6.320.3 Field Documentation**

- 6.320.3.1 **std::vector< decaf::lang::Pointer<BrokerId> >**  
**activemq::commands::DestinationInfo::brokerPath** [protected]
- 6.320.3.2 **Pointer<ConnectionId> activemq::commands::DestinationInfo::connectionId**  
[protected]
- 6.320.3.3 **Pointer<ActiveMQDestination> ac-**  
**tivemq::commands::DestinationInfo::destination**  
[protected]
- 6.320.3.4 **const unsigned char activemq::commands::DestinationInfo::ID\_-**  
**DESTINATIONINFO = 8** [static]
- 6.320.3.5 **unsigned char activemq::commands::DestinationInfo::operationType**  
[protected]
- 6.320.3.6 **long long activemq::commands::DestinationInfo::timeout**  
[protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**DestinationInfo.h**

## 6.321 **activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1784).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/DestinationInfo
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller**:

**Public Member Functions**

- **DestinationInfoMarshaller ()**
- **virtual ~DestinationInfoMarshaller ()**

- virtual **commands::DataStructure \* createObject ()** const

*Creates a new instance of this marshalable type.*

- virtual unsigned char **getDataStructureType ()** const

*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)**  
**throw ( decaf::io::IOException )**

*Un-marshall an object instance from the data input stream.*

- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn)** throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut)** throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.321.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1784).  
NOTE!: This file is auto generated - do not modify! if you need to make a change,  
please see the Java Classes in the activemq-openwire-generator module

### 6.321.2 Constructor & Destructor Documentation

6.321.2.1 `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::DestinationInfoMarshaller ( ) [inline]`

6.321.2.2 `virtual activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::~~DestinationInfoMarshaller ( ) [inline, virtual]`

### 6.321.3 Member Function Documentation

6.321.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.321.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.321.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 808).

**6.321.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 809).

**6.321.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 810).

```
6.321.3.6 virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 811).

```
6.321.3.7 virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 813).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/DestinationInfoMarshaller.h

## 6.322 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1789).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller:

### Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.322.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1789).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.322.2 Constructor & Destructor Documentation

6.322.2.1 **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::DestinationInfoMarshaller**  
 ( ) [inline]

6.322.2.2 **virtual activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::~~DestinationInfoMarshaller**  
 ( ) [inline, virtual]

### 6.322.3 Member Function Documentation

6.322.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::createObject** ( ) **const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.322.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::getDataStructureType**  
 ( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.322.3.3 **virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::looseMarshal**  
 ( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.



## 6.322 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller

### Class Reference 1793

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 772).

**6.322.3.4** virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::looseUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 774).

**6.322.3.5** virtual int activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::tightMarshal1 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 775).

```
6.322.3.6 virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 776).

```
6.322.3.7 virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## 6.323 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller Class Reference 1795

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**DestinationInfoMarshaller.h**

## 6.323 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1793).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller**:

### Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.323.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1793).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.323.2 Constructor & Destructor Documentation

- 6.323.2.1 **activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::DestinationInfoMarshaller**  
 ( ) [inline]
- 6.323.2.2 **virtual activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::~~DestinationInfoMarshaller**  
 ( ) [inline, virtual]

### 6.323.3 Member Function Documentation

- 6.323.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

- 6.323.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

## 6.323 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller

### Class Reference 1797

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.323.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 780).

**6.323.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 781).

```

6.323.3.5 virtual int activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 782).

```

6.323.3.6 virtual void activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 783).

## 6.324 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller Class Reference 1799

6.323.3.7 virtual void activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**DestinationInfoMarshaller.h**

## 6.324 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1797).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/DestinationInfoMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller:

### Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.324.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1797).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.324.2 Constructor & Destructor Documentation

6.324.2.1 **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::DestinationInfoMarshaller**  
( ) [inline]

6.324.2.2 **virtual activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::~~DestinationInfoMarshaller**  
( ) [inline, virtual]

### 6.324.3 Member Function Documentation

6.324.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.



### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.324.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.324.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 787).

**6.324.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 788).

```
6.324.3.5 virtual int activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 789).

```
6.324.3.6 virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

## 6.325 activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller Class Reference 1803

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 790).

**6.324.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 792).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**DestinationInfoMarshaller.h**

## 6.325 activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1801).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/DestinationInfoMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller**:

## Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.325.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1801).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.325.2 Constructor & Destructor Documentation

6.325.2.1 `activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::DestinationInfoMarshaller ( ) [inline]`

6.325.2.2 `virtual activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::~~DestinationInfoMarshaller ( ) [inline, virtual]`

## 6.325.3 Member Function Documentation

6.325.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.325.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.325.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 801).

**6.325.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 802).

**6.325.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.325 activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller

### Class Reference

1807

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 803).

```
6.325.3.6 virtual void activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataOutputStream * dataOut,
 utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 804).

```
6.325.3.7 virtual void activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
 dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
 * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 806).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/DestinationInfoMarshaller.h`

## 6.326 `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1806).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/DestinationInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller`:

### Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )



*Write a object instance to data output stream.*

### 6.326.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1806).  
NOTE!: This file is auto generated - do not modify! if you need to make a change,  
please see the Java Classes in the activemq-openwire-generator module

### 6.326.2 Constructor & Destructor Documentation

**6.326.2.1** `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::DestinationInfoMarshaller  
( ) [inline]`

**6.326.2.2** `virtual activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::~~DestinationInfoMarshaller  
( ) [inline, virtual]`

### 6.326.3 Member Function Documentation

**6.326.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.326.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::getDataStructureType  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.326.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::looseMarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 794).

**6.326.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 795).

**6.326.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 796).

**6.326.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 797).

**6.326.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**DestinationInfoMarshaller.h**

**6.327 activemq::cmsutil::DestinationResolver Class Reference**

Resolves a CMS destination name to a *Destination*.

```
#include <src/main/activemq/cmsutil/DestinationResolver.h>
```

Inheritance diagram for **activemq::cmsutil::DestinationResolver**:

**Public Member Functions**

- virtual **~DestinationResolver** ()
- virtual void **init** (**ResourceLifecycleManager** \*mgr)=0  
*Initializes this destination resolver for use.*
- virtual void **destroy** ()=0  
*Destroys any allocated resources.*
- virtual **cms::Destination** \* **resolveDestinationName** (**cms::Session** \*session, const std::string &destName, bool pubSubDomain)=0 throw ( cms::CMSException )  
*Resolves the given name to a destination.*

**6.327.1 Detailed Description**

Resolves a CMS destination name to a *Destination*.

## 6.327.2 Constructor & Destructor Documentation

**6.327.2.1** `virtual activemq::cmsutil::DestinationResolver::~~DestinationResolver ( )`  
`[inline, virtual]`

## 6.327.3 Member Function Documentation

**6.327.3.1** `virtual void activemq::cmsutil::DestinationResolver::destroy ( )` `[pure virtual]`

Destroys any allocated resources.

Implemented in `activemq::cmsutil::DynamicDestinationResolver` (p. 1879).

**6.327.3.2** `virtual void activemq::cmsutil::DestinationResolver::init (`  
`ResourceLifecycleManager * mgr )` `[pure virtual]`

Initializes this destination resolver for use.

Ensures that any previously allocated resources are first destroyed (e.g. calls `destroy()` (p. 1811)).

### Parameters

<i>mgr</i>	the resource lifecycle manager.
------------	---------------------------------

Implemented in `activemq::cmsutil::DynamicDestinationResolver` (p. 1879).

**6.327.3.3** `virtual cms::Destination* activemq::cmsutil::DestinationResolver::resolveDestinationName`  
`( cms::Session * session, const std::string & destName, bool pubSubDomain )`  
`throw ( cms::CMSException )` `[pure virtual]`

Resolves the given name to a destination.

If `pubSubDomain` is true, a topic will be returned, otherwise a queue will be returned.

### Parameters

<i>session</i>	the session for which to retrieve resolve the destination.
<i>destName</i>	the name to be resolved.
<i>pubSubDomain</i>	If true, the name will be resolved to a Topic, otherwise a Queue.

### Returns

the resolved destination

### Exceptions

<i>cms::CMSException</i> (p. 1190)	if resolution failed.
---------------------------------------	-----------------------

Implemented in **activemq::cmsutil::DynamicDestinationResolver** (p. 1880).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**DestinationResolver.h**

## 6.328 activemq::commands::DiscoveryEvent Class Reference

```
#include <src/main/activemq/commands/DiscoveryEvent.h>
```

Inheritance diagram for activemq::commands::DiscoveryEvent:

### Public Member Functions

- **DiscoveryEvent** ()
- virtual **~DiscoveryEvent** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **DiscoveryEvent \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getServiceName** () const
- virtual std::string & **getServiceName** ()
- virtual void **setServiceName** (const std::string &serviceName)
- virtual const std::string & **getBrokerName** () const
- virtual std::string & **getBrokerName** ()
- virtual void **setBrokerName** (const std::string &brokerName)

### Static Public Attributes

- static const unsigned char **ID\_DISCOVERYEVENT** = 40

### Protected Attributes

- std::string **serviceName**
- std::string **brokerName**

### 6.328.1 Constructor & Destructor Documentation

6.328.1.1 `activemq::commands::DiscoveryEvent::DiscoveryEvent ( )`

6.328.1.2 `virtual activemq::commands::DiscoveryEvent::~DiscoveryEvent ( ) [virtual]`

### 6.328.2 Member Function Documentation

6.328.2.1 `virtual DiscoveryEvent* activemq::commands::DiscoveryEvent::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1714).

6.328.2.2 `virtual void activemq::commands::DiscoveryEvent::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Implements `activemq::commands::DataStructure` (p. 1715).

6.328.2.3 `virtual bool activemq::commands::DiscoveryEvent::equals ( const DataStructure * value ) const [virtual]`

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1716).

6.328.2.4 **virtual const std::string& activemq::commands::DiscoveryEvent::getBrokerName ( )**  
**const** [virtual]

6.328.2.5 **virtual std::string& activemq::commands::DiscoveryEvent::getBrokerName ( )**  
[virtual]

6.328.2.6 **virtual unsigned char activemq::commands::DiscoveryEvent::getDataStructureType (**  
**) const** [virtual]

Get the unique identifier that this object and its own Marshaler share.

### Returns

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

6.328.2.7 **virtual std::string& activemq::commands::DiscoveryEvent::getServiceName ( )**  
[virtual]

6.328.2.8 **virtual const std::string& activemq::commands::DiscoveryEvent::getServiceName ( )**  
**const** [virtual]

6.328.2.9 **virtual void activemq::commands::DiscoveryEvent::setBrokerName ( const std::string**  
**& brokerName )** [virtual]

6.328.2.10 **virtual void activemq::commands::DiscoveryEvent::setServiceName ( const**  
**std::string & serviceName )** [virtual]

6.328.2.11 **virtual std::string activemq::commands::DiscoveryEvent::toString ( ) const**  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 841).



### 6.328.3 Field Documentation

6.328.3.1 **std::string activemq::commands::DiscoveryEvent::brokerName**  
[protected]

6.328.3.2 **const unsigned char activemq::commands::DiscoveryEvent::ID\_-  
DISCOVERYEVENT = 40** [static]

6.328.3.3 **std::string activemq::commands::DiscoveryEvent::serviceName**  
[protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**DiscoveryEvent.h**

## 6.329 **activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller** Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1815).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/DiscoveryEventMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller**:

### Public Member Functions

- **DiscoveryEventMarshaller ()**
- virtual **~DiscoveryEventMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)**  
**throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** **throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.329.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p.1815).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.329.2 Constructor & Destructor Documentation

- 6.329.2.1 **activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::DiscoveryEventMarshaller**  
 ( ) [inline]
- 6.329.2.2 **virtual activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller**  
 ( ) [inline, virtual]

### 6.329.3 Member Function Documentation

- 6.329.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1661).

- 6.329.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

## Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.329.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.329.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.329.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
`[virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1690).

**6.329.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
`[virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1698).

## 6.330 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller

### Class Reference

1821

6.329.3.7 virtual void activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**DiscoveryEventMarshaller.h**

## 6.330 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller

### Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1819).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryEventMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller:

#### Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual ~**DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.330.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p.1819).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.330.2 Constructor & Destructor Documentation

6.330.2.1 **activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::DiscoveryEventMarshaller**  
( ) [inline]

6.330.2.2 **virtual activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller**  
( ) [inline, virtual]

### 6.330.3 Member Function Documentation

6.330.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.330.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.330.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.330.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.330.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.330.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



### 6.331 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller

#### Class Reference

1825

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

6.330.3.7 virtual void **activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**DiscoveryEventMarshaller.h**

### 6.331 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller

#### Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1823).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller**:

#### Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.331.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1823).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.331.2 Constructor & Destructor Documentation

- 6.331.2.1 **activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::DiscoveryEventMarshaller**  
( ) [inline]
- 6.331.2.2 **virtual activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller**  
( ) [inline, virtual]

### 6.331.3 Member Function Documentation

- 6.331.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.331.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.331.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.331.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.331.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.331.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.332 activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller

### Class Reference

1829

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

6.331.3.7 virtual void **activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**DiscoveryEventMarshaller.h**

## 6.332 activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller

### Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1827).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller**:

#### Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.332.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1827).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.332.2 Constructor & Destructor Documentation

- 6.332.2.1 **activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::DiscoveryEventMarshaller**  
( ) [inline]
- 6.332.2.2 **virtual activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller**  
( ) [inline, virtual]

### 6.332.3 Member Function Documentation

- 6.332.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

## 6.332 activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller

### Class Reference

1831

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.332.3.2** virtual unsigned char activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.332.3.3** virtual void activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::looseMarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.332.3.4** virtual void activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::looseUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.332.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.332.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



### 6.333 activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller

#### Class Reference

1833

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

6.332.3.7 virtual void **activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**DiscoveryEventMarshaller.h**

### 6.333 activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller

#### Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1831).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller**:

#### Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual ~**DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.333.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1831).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.333.2 Constructor & Destructor Documentation

- 6.333.2.1 **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::DiscoveryEventMarshaller**  
( ) [inline]
- 6.333.2.2 **virtual activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller**  
( ) [inline, virtual]

### 6.333.3 Member Function Documentation

- 6.333.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.333.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::getDataStructureType ( ) const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

## Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.333.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.333.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.333.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.333.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.334 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller

### Class Reference

1837

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

6.333.3.7 virtual void **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**DiscoveryEventMarshaller.h**

## 6.334 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller

### Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1835).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller**:

#### Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.334.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1835).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.334.2 Constructor & Destructor Documentation

- 6.334.2.1 **activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::DiscoveryEventMarshaller**  
( ) [inline]
- 6.334.2.2 **virtual activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller**  
( ) [inline, virtual]

### 6.334.3 Member Function Documentation

- 6.334.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

### 6.334 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller

#### Class Reference

1839

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.334.3.2** virtual unsigned char activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.334.3.3** virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::looseMarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.334.3.4** virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::looseUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.334.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.334.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.334.3.7 virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
  * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h

## 6.335 activemq::core::DispatchData Class Reference

Simple POCO that contains the information necessary to route a message to a specified consumer.

```
#include <src/main/activemq/core/DispatchData.h>
```

#### Public Member Functions

- **DispatchData** ()
- **DispatchData** (const decaf::lang::Pointer< commands::ConsumerId > &consumer, const decaf::lang::Pointer< commands::Message > &message)
- const decaf::lang::Pointer< commands::ConsumerId > & getConsumerId ()
- const decaf::lang::Pointer< commands::Message > & getMessage ()

#### 6.335.1 Detailed Description

Simple POCO that contains the information necessary to route a message to a specified consumer.

### 6.335.2 Constructor & Destructor Documentation

6.335.2.1 `activemq::core::DispatchData::DispatchData ( )` `[inline]`

6.335.2.2 `activemq::core::DispatchData::DispatchData ( const decaf::lang::Pointer< commands::ConsumerId > & consumer, const decaf::lang::Pointer< commands::Message > & message )` `[inline]`

### 6.335.3 Member Function Documentation

6.335.3.1 `const decaf::lang::Pointer< commands::ConsumerId > & activemq::core::DispatchData::getConsumerId ( )` `[inline]`

6.335.3.2 `const decaf::lang::Pointer< commands::Message > & activemq::core::DispatchData::getMessage ( )` `[inline]`

The documentation for this class was generated from the following file:

- `src/main/activemq/core/DispatchData.h`

## 6.336 activemq::core::Dispatcher Class Reference

Interface for an object responsible for dispatching messages to consumers.

```
#include <src/main/activemq/core/Dispatcher.h>
```

Inheritance diagram for `activemq::core::Dispatcher`:

### Public Member Functions

- `virtual ~Dispatcher ( )`
- `virtual void dispatch (const Pointer< MessageDispatch > &message)=0`

*Dispatches a message to a particular consumer.*

#### 6.336.1 Detailed Description

Interface for an object responsible for dispatching messages to consumers.

### 6.336.2 Constructor & Destructor Documentation

6.336.2.1 virtual activemq::core::Dispatcher::~~Dispatcher ( ) [inline, virtual]

### 6.336.3 Member Function Documentation

6.336.3.1 virtual void activemq::core::Dispatcher::dispatch ( const Pointer< MessageDispatch > & message ) [pure virtual]

Dispatches a message to a particular consumer.

#### Parameters

<i>message</i>	- the message to be dispatched.
----------------	---------------------------------

Implemented in **activemq::core::ActiveMQConsumer** (p. 305), and **activemq::core::ActiveMQSession** (p. 525).

The documentation for this class was generated from the following file:

- src/main/activemq/core/Dispatcher.h

## 6.337 decaf::lang::Double Class Reference

```
#include <src/main/decaf/lang/Double.h>
```

Inheritance diagram for decaf::lang::Double:

### Public Member Functions

- **Double** (double value)
- **Double** (const std::string &value) throw ( exceptions::NumberFormatException )
- virtual ~**Double** ()
- virtual int **compareTo** (const **Double** &d) const

*Compares this **Double** (p. 1841) instance with another.*

- bool **equals** (const **Double** &d) const
- virtual bool **operator==** (const **Double** &d) const

*Compares equality between this object and the one passed.*

- virtual bool **operator<** (const **Double** &d) const

*Compares this object to another and returns true if this object is considered to be less than the one passed.*

- virtual int **compareTo** (const double &d) const  
*Compares this **Double** (p. 1841) instance with another.*
- bool **equals** (const double &d) const
- virtual bool **operator==** (const double &d) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const double &d) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- std::string **toString** () const
- virtual double **doubleValue** () const  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const  
*Answers the float value which the receiver represents.*
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual short **shortValue** () const  
*Answers the short value which the receiver represents.*
- virtual int **intValue** () const  
*Answers the int value which the receiver represents.*
- virtual long long **longValue** () const  
*Answers the long value which the receiver represents.*
- bool **isInfinite** () const
- bool **isNaN** () const

### Static Public Member Functions

- static int **compare** (double d1, double d2)  
*Compares the two specified double values.*
- static long long **doubleToLongBits** (double value)  
*Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout.*
- static long long **doubleToRawLongBits** (double value)  
*Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout, preserving Not-a-Number (NaN) values.*

- static bool **isInfinite** (double value)
- static bool **isNaN** (double value)
- static double **longBitsToDouble** (long long bits)  
*Returns the double value corresponding to a given bit representation.*
- static double **parseDouble** (const std::string value) throw ( exceptions::NumberFormatException )  
*Returns a new double initialized to the value represented by the specified string, as performed by the valueOf method of class **Double** (p. 1841).*
- static std::string **toHexString** (double value)  
*Returns a hexadecimal string representation of the double argument.*
- static std::string **toString** (double value)  
*Returns a string representation of the double argument.*
- static **Double valueOf** (double value)  
*Returns a **Double** (p. 1841) instance representing the specified double value.*
- static **Double valueOf** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Returns a **Double** (p. 1841) instance that wraps a primitive double which is parsed from the string value passed.*

### Static Public Attributes

- static const int **SIZE** = 64  
*The size in bits of the primitive int type.*
- static const double **MAX\_VALUE**  
*The maximum value that the primitive type can hold.*
- static const double **MIN\_VALUE**  
*The minimum value that the primitive type can hold.*
- static const double **NaN**  
*Constant for the Not a **Number** (p. 2918) Value.*
- static const double **POSITIVE\_INFINITY**  
*Constant for Positive Infinity.*
- static const double **NEGATIVE\_INFINITY**  
*Constant for Negative Infinity.*

### 6.337.1 Constructor & Destructor Documentation

#### 6.337.1.1 `decaf::lang::Double::Double ( double value )`

##### Parameters

<i>value</i>	- the primitive type to wrap
--------------	------------------------------

#### 6.337.1.2 `decaf::lang::Double::Double ( const std::string & value ) throw ( exceptions::NumberFormatException )`

##### Parameters

<i>value</i>	- the string to convert to a primitive type to wrap
--------------	---

#### 6.337.1.3 `virtual decaf::lang::Double::~~Double ( ) [inline, virtual]`

### 6.337.2 Member Function Documentation

#### 6.337.2.1 `virtual unsigned char decaf::lang::Double::byteValue ( ) const [inline, virtual]`

Answers the byte value which the receiver represents.

##### Returns

byte the value of the receiver.

Reimplemented from `decaf::lang::Number` (p. 2919).

#### 6.337.2.2 `static int decaf::lang::Double::compare ( double d1, double d2 ) [static]`

Compares the two specified double values.

The sign of the integer value returned is the same as that of the integer that would be returned by the call: `new Double(d1).compareTo(new Double(d2))`

##### Parameters

<i>d1</i>	- the first double to compare
<i>d2</i>	- the second double to compare

##### Returns

the value 0 if *d1* is numerically equal to *d2*; a value less than 0 if *d1* is numerically less than *d2*; and a value greater than 0 if *d1* is numerically greater than *d2*.

**6.337.2.3** `virtual int decaf::lang::Double::compareTo ( const double & d ) const`  
[virtual]

Compares this **Double** (p. 1841) instance with another.

#### Parameters

<i>d</i>	- the <b>Double</b> (p. 1841) instance to be compared
----------	---

#### Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **double** > (p. 1249).

**6.337.2.4** `virtual int decaf::lang::Double::compareTo ( const Double & d ) const`  
[virtual]

Compares this **Double** (p. 1841) instance with another.

#### Parameters

<i>d</i>	- the <b>Double</b> (p. 1841) instance to be compared
----------	---

#### Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **Double** > (p. 1249).

**6.337.2.5** `static long long decaf::lang::Double::doubleToLongBits ( double value )`  
[static]

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout.

Bit 63 (the bit that is selected by the mask 0x8000000000000000L) represents the sign of the floating-point number. Bits 62-52 (the bits that are selected by the mask 0x7ff0000000000000L) represent the exponent. Bits 51-0 (the bits that are selected by the mask 0x000fffffffffffffL) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7ff0000000000000L. If the argument is negative infinity, the result is 0xfff0000000000000L. If the argument is NaN, the result is 0x7ff8000000000000L.

In all cases, the result is a long integer that, when given to the longBitsToDouble(long)

method, will produce a floating-point value the same as the argument to `doubleToLongBits` (except all NaN values are collapsed to a single "canonical" NaN value).

#### Parameters

<i>value</i>	- double to be converted
--------------	--------------------------

#### Returns

the long long bits that make up the double

**6.337.2.6** `static long long decaf::lang::Double::doubleToRawLongBits ( double value )`  
`[static]`

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout, preserving Not-a-Number (NaN) values.

Bit 63 (the bit that is selected by the mask `0x8000000000000000LL`) represents the sign of the floating-point number. Bits 62-52 (the bits that are selected by the mask `0x7ff0000000000000L`) represent the exponent. Bits 51-0 (the bits that are selected by the mask `0x000ffffffffffffL`) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is `0x7ff0000000000000LL`. If the argument is negative infinity, the result is `0xfff0000000000000LL`. If the argument is NaN, the result is the long integer representing the actual NaN value. Unlike the `doubleToLongBits` method, `doubleToRawLongBits` does not collapse all the bit patterns encoding a NaN to a single "canonical" NaN value.

In all cases, the result is a long integer that, when given to the `longBitsToDouble(long)` method, will produce a floating-point value the same as the argument to `doubleToRawLongBits`.

#### Parameters

<i>value</i>	- double to be converted
--------------	--------------------------

#### Returns

the long long bits that make up the double

**6.337.2.7** `virtual double decaf::lang::Double::doubleValue ( ) const` `[inline, virtual]`

Answers the double value which the receiver represents.

#### Returns

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2919).



**6.337.2.8** `bool decaf::lang::Double::equals ( const Double & d ) const` `[inline, virtual]`

#### Parameters

<code>d</code>	- the <b>Double</b> (p. 1841) object to compare against.
----------------	--

#### Returns

true if the two **Double** (p. 1841) Objects have the same value.

Implements **decaf::lang::Comparable**< **Double** > (p. 1250).

**6.337.2.9** `bool decaf::lang::Double::equals ( const double & d ) const` `[inline, virtual]`

#### Parameters

<code>d</code>	- the <b>Double</b> (p. 1841) object to compare against.
----------------	--

#### Returns

true if the two **Double** (p. 1841) Objects have the same value.

Implements **decaf::lang::Comparable**< **double** > (p. 1250).

**6.337.2.10** `virtual float decaf::lang::Double::floatValue ( ) const` `[inline, virtual]`

Answers the float value which the receiver represents.

#### Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2920).

**6.337.2.11** `virtual int decaf::lang::Double::intValue ( ) const` `[inline, virtual]`

Answers the int value which the receiver represents.

#### Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2920).

**6.337.2.12** `bool decaf::lang::Double::isInfinite ( ) const`

#### Returns

true if the double is equal to positive infinity.

**6.337.2.13** `static bool decaf::lang::Double::isInfinite ( double value ) [static]`

#### Parameters

<i>value</i>	- The double to check.
--------------	------------------------

#### Returns

true if the double is equal to infinity.

**6.337.2.14** `bool decaf::lang::Double::isNaN ( ) const`

#### Returns

true if the double is equal to NaN.

**6.337.2.15** `static bool decaf::lang::Double::isNaN ( double value ) [static]`

#### Parameters

<i>value</i>	- The double to check.
--------------	------------------------

#### Returns

true if the double is equal to NaN.

**6.337.2.16** `static double decaf::lang::Double::longBitsToDouble ( long long bits ) [static]`

Returns the double value corresponding to a given bit representation.

The argument is considered to be a representation of a floating-point value according to the IEEE 754 floating-point "double format" bit layout.

If the argument is 0x7ff0000000000000L, the result is positive infinity. If the argument is 0xfff0000000000000L, the result is negative infinity. If the argument is any value in the range 0x7ff0000000000001L through 0x7fffffffffffffffL or in the range 0xfff0000000000001L through 0xfffffffffffffffL, the result is a NaN. No IEEE 754 floating-point operation provided by C++ can distinguish between two NaN values of the same type with different bit patterns. Distinct values of NaN are only distinguishable by use of the **Double.doubleToRawLongBits** (p. 1846) method.

#### Parameters

<i>bits</i>	- the long long bits to convert to double
-------------	---

#### Returns

the double converted from the bits

**6.337.2.17** `virtual long long decaf::lang::Double::longValue ( ) const` `[inline, virtual]`

Answers the long value which the receiver represents.

#### Returns

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2920).

**6.337.2.18** `virtual bool decaf::lang::Double::operator< ( const Double & d ) const` `[inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>d</i> - the value to be compared to this one.
--

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Double >** (p. 1250).

**6.337.2.19** `virtual bool decaf::lang::Double::operator< ( const double & d ) const` `[inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>d</i> - the value to be compared to this one.
--

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< double >** (p. 1250).

**6.337.2.20** `virtual bool decaf::lang::Double::operator== ( const Double & d ) const` `[inline, virtual]`

Compares equality between this object and the one passed.

**Parameters**

<i>d</i>	- the value to be compared to this one.
----------	---

**Returns**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Double** > (p. 1250).

**6.337.2.21** `virtual bool decaf::lang::Double::operator== ( const double & d ) const`  
`[inline, virtual]`

Compares equality between this object and the one passed.

**Parameters**

<i>d</i>	- the value to be compared to this one.
----------	---

**Returns**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **double** > (p. 1250).

**6.337.2.22** `static double decaf::lang::Double::parseDouble ( const std::string value ) throw (`  
`exceptions::NumberFormatException )` `[static]`

Returns a new double initialized to the value represented by the specified string, as performed by the `valueOf` method of class **Double** (p. 1841).

**Parameters**

<i>value</i>	- The string to parse to an double
--------------	------------------------------------

**Returns**

a double parsed from the passed string

**Exceptions**

<i>NumberFormatException</i>	
------------------------------	--

**6.337.2.23** `virtual short decaf::lang::Double::shortValue ( ) const` `[inline,`  
`virtual]`

Answers the short value which the receiver represents.

**Returns**

short the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2920).

**6.337.2.24 static std::string decaf::lang::Double::toHexString ( double value ) [static]**

Returns a hexadecimal string representation of the double argument.

All characters mentioned below are ASCII characters.

\* If the argument is NaN, the result is the string "NaN". \* Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the string "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the string "0x0.0p0"; thus, negative zero produces the result "-0x0.0p0" and positive zero produces the result "0x0.0p0". o If m is a double value with a normalized representation, substrings are used to represent the significand and exponent fields. The significand is represented by the characters "0x1." followed by a lowercase hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed unless all the digits are zero, in which case a single zero is used. Next, the exponent is represented by "p" followed by a decimal string of the unbiased exponent as if produced by a call to **Integer.toString** (p. 2157) on the exponent value. o If m is a double value with a subnormal representation, the significand is represented by the characters "0x0." followed by a hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed. Next, the exponent is represented by "p-126". Note that there must be at least one nonzero digit in a subnormal significand.

**Parameters**

<i>value</i>	- The double to convert to a string
--------------	-------------------------------------

**Returns**

the Hex formatted double string.

**6.337.2.25 std::string decaf::lang::Double::toString ( ) const****Returns**

this **Double** (p. 1841) Object as a **String** (p. 3775) Representation

**6.337.2.26 static std::string decaf::lang::Double::toString ( double value ) [static]**

Returns a string representation of the double argument.

All characters mentioned below are ASCII characters.

If the argument is NaN, the result is the string "NaN". Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m:

- o If m is infinity, it is represented by the characters "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity".
- o If m is zero, it is represented by the characters "0.0"; thus, negative zero produces the result "-0.0" and positive zero produces the result "0.0".
- o If m is greater than or equal to 10<sup>-3</sup> but less than 10<sup>7</sup>, then it is represented as the integer part of m, in decimal form with no leading zeroes, followed by '.', followed by one or more decimal digits representing the fractional part of m.
- o If m is less than 10<sup>-3</sup> or greater than or equal to 10<sup>7</sup>, then it is represented in so-called "computerized scientific notation." Let n be the unique integer such that 10<sup>n</sup> ≤ m < 10<sup>n+1</sup>; then let a be the mathematically exact quotient of m and 10<sup>n</sup> so that 1 ≤ a < 10. The magnitude is then represented as the integer part of a, as a single decimal digit, followed by '.', followed by decimal digits representing the fractional part of a, followed by the letter 'E', followed by a representation of n as a decimal integer, as produced by the method **Integer.toString(int)** (p. 2158).

#### Parameters

<i>value</i>	- The double to convert to a string
--------------	-------------------------------------

#### Returns

the formatted double string.

**6.337.2.27** `static Double decaf::lang::Double::valueOf ( double value ) [static]`

Returns a **Double** (p. 1841) instance representing the specified double value.

#### Parameters

<i>value</i>	- double to wrap
--------------	------------------

#### Returns

new **Double** (p. 1841) instance wrapping the primitive value

**6.337.2.28** `static Double decaf::lang::Double::valueOf ( const std::string & value ) throw ( exceptions::NumberFormatException ) [static]`

Returns a **Double** (p. 1841) instance that wraps a primitive double which is parsed from the string value passed.

#### Parameters

<i>value</i>	- the string to parse
--------------	-----------------------

Returns

a new **Double** (p. 1841) instance wrapping the double parsed from value

Exceptions

<i>NumberFormatException</i>	on error.
------------------------------	-----------

6.337.3 Field Documentation

6.337.3.1 `const double decaf::lang::Double::MAX_VALUE` [static]

The maximum value that the primitive type can hold.

6.337.3.2 `const double decaf::lang::Double::MIN_VALUE` [static]

The minimum value that the primitive type can hold.

6.337.3.3 `const double decaf::lang::Double::NaN` [static]

Constant for the Not a **Number** (p. 2918) Value.

6.337.3.4 `const double decaf::lang::Double::NEGATIVE_INFINITY` [static]

Constant for Negative Infinitiy.

6.337.3.5 `const double decaf::lang::Double::POSITIVE_INFINITY` [static]

Constant for Positive Infinity.

6.337.3.6 `const int decaf::lang::Double::SIZE = 64` [static]

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Double.h**

6.338 decaf::internal::nio::DoubleArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/DoubleArrayBuffer.h>
```

Inheritance diagram for decaf::internal::nio::DoubleArrayBuffer:

## Public Member Functions

- **DoubleArrayBuffer** (int capacity, bool readOnly=false) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Creates a **DoubleArrayBuffer** (p. 1853) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **DoubleArrayBuffer** (double \*array, int size, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a **DoubleArrayBuffer** (p. 1853) object that wraps the given array.*

- **DoubleArrayBuffer** (const decaf::lang::Pointer< ByteArrayAdapter > &array, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.*

- **DoubleArrayBuffer** (const **DoubleArrayBuffer** &other)

*Create a **DoubleArrayBuffer** (p. 1853) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*

- virtual ~**DoubleArrayBuffer** ()
- virtual double \* **array** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

*Returns the double array that backs this buffer (optional operation).*

*Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.*

*Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.*

**Returns**

*the array that backs this **Buffer** (p. 936).*

**Exceptions**

<b>ReadOnlyBufferException</b> (p. 3260)	<i>if this <b>Buffer</b> (p. 936) is read only.</i>
UnsupportedOperationException	<i>if the underlying store has no array.</i>

- virtual int **arrayOffset** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*

*Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.*

**Returns**

*The offset into the backing array where index zero starts.*



**Exceptions**

<b>ReadOnlyBufferException</b> (p. 3260)	if this <i>Buffer</i> (p. 936) is read only.
UnsupportedOperationException	if the underlying store has no array.

- virtual *DoubleBuffer* \* **asReadOnlyBuffer** () const

*Creates a new, read-only double buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.*

*If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.*

*The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.*

**Returns**

*The new, read-only double buffer which the caller then owns.*

- virtual *DoubleBuffer* & **compact** () throw ( decaf::nio::ReadOnlyBufferException )

*Compacts this buffer.*

*The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 941) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 940) - 1 is copied to index  $n = \text{limit}()$  (p. 940) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.*

*The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.*

**Returns**

*a reference to this **DoubleBuffer** (p. 1865).*

**Exceptions**

<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only.
---	------------------------------

- virtual *DoubleBuffer* \* **duplicate** ()

*Creates a new double buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns**

*a new double **Buffer** (p. 936) which the caller owns.*

- virtual double **get** () throw ( decaf::nio::BufferUnderflowException )

*Relative get method.*

*Reads the value at this buffer's current position, and then increments the position.*

**Returns**

*the double at the current position.*

**Exceptions**

<b>BufferUnderflowException</b> (p. 967)	if there no more data to return.
---	----------------------------------

- virtual double **get** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Absolute get method.*

*Reads the value at the given index.*

**Parameters**

index	The index in the <b>Buffer</b> (p. 936) where the double is to be read.
-------	---

**Returns**

*the double that is located at the given index.*

**Exceptions**

<b>IndexOutOfBoundsException</b>	if index is not smaller than the buffer's limit
----------------------------------	---

- virtual bool **hasArray** () const

*Tells whether or not this buffer is backed by an accessible double array.*

*If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.*

**Returns**

*true if, and only if, this buffer is backed by an array and is not read-only*

- virtual bool **isReadOnly** () const

*Tells whether or not this buffer is read-only.*

**Returns**

*true if, and only if, this buffer is read-only.*

- virtual DoubleBuffer & **put** (double value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes the given doubles into this buffer at the current position, and then increments the position.*

**Parameters**

value	The doubles value to be written.
-------	----------------------------------

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 964)	<i>if this buffer's current position is not smaller than its limit.</i>
<b>ReadOnlyBufferException</b> (p. 3260)	<i>if this buffer is read-only.</i>

- virtual DoubleBuffer & **put** (int index, double value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes the given doubles into this buffer at the given index.*

**Parameters**

index	<i>The position in the <b>Buffer</b> (p. 936) to write the data.</i>
value	<i>The doubles to write.</i>

**Returns**

*a reference to this buffer*

**Exceptions**

IndexOutOfBoundsException	<i>if index greater than the buffer's limit minus the size of the type being written, or the index is negative.</i>
<b>ReadOnlyBufferException</b> (p. 3260)	<i>if this buffer is read-only.</i>

- virtual DoubleBuffer \* **slice** () const

*Creates a new **DoubleBuffer** (p. 1865) whose content is a shared subsequence of this buffer's content.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns**

*the newly create **DoubleBuffer** (p. 1865) which the caller owns.*

**Protected Member Functions**

- virtual void **setReadOnly** (bool value)

*Sets this **DoubleArrayBuffer** (p. 1853) as Read-Only or not Read-Only.*

### 6.338.1 Constructor & Destructor Documentation

**6.338.1.1** `decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer`  
 ( `int capacity`, `bool readOnly = false` ) `throw (`  
`decaf::lang::exceptions::IllegalArgumentException` `)`

Creates a **DoubleArrayBuffer** (p. 1853) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

#### Parameters

<i>size</i>	The size of the array, this is the limit we read and write to.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>IllegalArgumentException</i>	if the capacity value is negative.
---------------------------------	------------------------------------

**6.338.1.2** `decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer` ( `double`  
`* array`, `int size`, `int offset`, `int length`, `bool readOnly = false`  
`) throw ( decaf::lang::exceptions::NullPointerException,`  
`decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a **DoubleArrayBuffer** (p. 1853) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

#### Parameters

<i>array</i>	The actual array to wrap.
<i>size</i>	The size of the given array.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if offset is greater than array capacity.

```
6.338.1.3 decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer
( const decaf::lang::Pointer< ByteArrayAdapter >
& array, int offset, int length, bool readOnly = false )
throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IndexOutOfBoundsException )
```

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **DoubleArrayBuffer** (p. 1853) will be that of the remaining capacity of the passed buffer.

#### Parameters

<i>array</i>	The ByteArrayAdapter to wrap.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>NullPointerException</i>	if array is NULL
<i>IndexOutOfBoundsException</i>	if offset + length is greater than array size.

```
6.338.1.4 decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer ( const
DoubleArrayBuffer & other )
```

Create a **DoubleArrayBuffer** (p. 1853) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.

#### Parameters

<i>other</i>	The <b>DoubleArrayBuffer</b> (p. 1853) this one is to mirror.
--------------	---

```
6.338.1.5 virtual decaf::internal::nio::DoubleArrayBuffer::~~DoubleArrayBuffer ( )
[virtual]
```

### 6.338.2 Member Function Documentation

```
6.338.2.1 virtual double* decaf::internal::nio::DoubleArrayBuffer::array ( ) throw
( decaf::lang::exceptions::UnsupportedOperationException,
decaf::nio::ReadOnlyBufferException ) [virtual]
```

Returns the double array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

the array that backs this **Buffer** (p. 936).

### Exceptions

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<i><b>UnsupportedOperationException</b></i>	if the underlying store has no array.

Implements **decaf::nio::DoubleBuffer** (p. 1868).

**6.338.2.2** `virtual int decaf::internal::nio::DoubleArrayBuffer::arrayOffset ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )` [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

The offset into the backing array where index zero starts.

### Exceptions

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<i><b>UnsupportedOperationException</b></i>	if the underlying store has no array.

Implements **decaf::nio::DoubleBuffer** (p. 1869).

**6.338.2.3** `virtual DoubleBuffer* decaf::internal::nio::DoubleArrayBuffer::asReadOnlyBuffer ( ) const` [virtual]

Creates a new, read-only double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and

will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

### Returns

The new, read-only double buffer which the caller then owns.

Implements **decaf::nio::DoubleBuffer** (p. 1869).

**6.338.2.4** `virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::compact ( ) throw ( decaf::nio::ReadOnlyBufferException )` [virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 941) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 940) - 1 is copied to index  $n = \text{limit}()$  (p. 940) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

### Returns

a reference to this **DoubleBuffer** (p. 1865).

### Exceptions

<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only.
---	------------------------------

Implements **decaf::nio::DoubleBuffer** (p. 1870).

**6.338.2.5** `virtual DoubleBuffer* decaf::internal::nio::DoubleArrayBuffer::duplicate ( )` [virtual]

Creates a new double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

a new double **Buffer** (p. 936) which the caller owns.

Implements **decaf::nio::DoubleBuffer** (p. 1870).

**6.338.2.6** virtual double decaf::internal::nio::DoubleArrayBuffer::get ( ) throw ( decaf::nio::BufferUnderflowException ) [virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

### Returns

the double at the current position.

### Exceptions

<i>BufferUnderflowException</i> (p. 967)	if there no more data to return.
---	----------------------------------

Implements **decaf::nio::DoubleBuffer** (p. 1872).

**6.338.2.7** virtual double decaf::internal::nio::DoubleArrayBuffer::get ( int *index* ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Absolute get method.

Reads the value at the given index.

### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the double is to be read.
--------------	---

### Returns

the double that is located at the given index.

### Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit
----------------------------------	---

Implements **decaf::nio::DoubleBuffer** (p. 1871).



**6.338.2.8** `virtual bool decaf::internal::nio::DoubleArrayBuffer::hasArray ( ) const`  
`[inline, virtual]`

Tells whether or not this buffer is backed by an accessible double array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

#### Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::DoubleBuffer** (p. 1873).

**6.338.2.9** `virtual bool decaf::internal::nio::DoubleArrayBuffer::isReadOnly ( ) const`  
`[inline, virtual]`

Tells whether or not this buffer is read-only.

#### Returns

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 940).

**6.338.2.10** `virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::put ( int index, double value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )` `[virtual]`

Writes the given doubles into this buffer at the given index.

#### Parameters

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data.
<i>value</i>	The doubles to write.

#### Returns

a reference to this buffer

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or the index is negative.
<i>ReadOnlyBufferException</i> (p. 3260)	if this buffer is read-only.

Implements **decaf::nio::DoubleBuffer** (p. 1874).

**6.338.2.11** `virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::put ( double value ) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )` [virtual]

Writes the given doubles into this buffer at the current position, and then increments the position.

#### Parameters

<i>value</i>	The doubles value to be written.
--------------	----------------------------------

#### Returns

a reference to this buffer.

#### Exceptions

<b><i>BufferOverflowException</i></b> (p. 964)	if this buffer's current position is not smaller than its limit.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.

Implements **decaf::nio::DoubleBuffer** (p. 1873).

**6.338.2.12** `virtual void decaf::internal::nio::DoubleArrayBuffer::setReadOnly ( bool value )`  
[inline, protected, virtual]

Sets this **DoubleArrayBuffer** (p. 1853) as Read-Only or not Read-Only.

#### Parameters

<i>value</i>	Boolean value, true if this buffer is to be read-only, false otherwise.
--------------	---

**6.338.2.13** `virtual DoubleBuffer* decaf::internal::nio::DoubleArrayBuffer::slice ( ) const`  
[virtual]

Creates a new **DoubleBuffer** (p. 1865) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns**

the newly create **DoubleBuffer** (p. 1865) which the caller owns.

Implements **decaf::nio::DoubleBuffer** (p. 1876).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**DoubleArrayBuffer.h**

**6.339 decaf::nio::DoubleBuffer Class Reference**

This class defines four categories of operations upon double buffers:

```
#include <src/main/decaf/nio/DoubleBuffer.h>
```

Inheritance diagram for decaf::nio::DoubleBuffer:

**Public Member Functions**

- virtual **~DoubleBuffer** ()
- virtual std::string **toString** () const
- virtual double \* **array** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )  
*Returns the double array that backs this buffer (optional operation).*
- virtual int **arrayOffset** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )  
*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*
- virtual **DoubleBuffer** \* **asReadOnlyBuffer** () const =0  
*Creates a new, read-only double buffer that shares this buffer's content.*
- virtual **DoubleBuffer** & **compact** ()=0 throw ( ReadOnlyBufferException )  
*Compacts this buffer.*
- virtual **DoubleBuffer** \* **duplicate** ()=0  
*Creates a new double buffer that shares this buffer's content.*
- virtual double **get** ()=0 throw ( BufferUnderflowException )  
*Relative get method.*
- virtual double **get** (int index) const =0 throw ( lang::exceptions::IndexOutOfBoundsException )  
*Absolute get method.*

- **DoubleBuffer & get** (std::vector< double > buffer) throw ( BufferUnderflowException )

*Relative bulk get method.*

- **DoubleBuffer & get** (double \*buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*Relative bulk get method.*

- virtual bool **hasArray** () const =0

*Tells whether or not this buffer is backed by an accessible double array.*

- **DoubleBuffer & put** (**DoubleBuffer** &src) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )

*This method transfers the doubles remaining in the given source buffer into this buffer.*

- **DoubleBuffer & put** (const double \*buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*This method transfers doubles into this buffer from the given source array.*

- **DoubleBuffer & put** (std::vector< double > &buffer) throw ( BufferOverflowException, ReadOnlyBufferException )

*This method transfers the entire content of the given source doubles array into this buffer.*

- virtual **DoubleBuffer & put** (double value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )

*Writes the given doubles into this buffer at the current position, and then increments the position.*

- virtual **DoubleBuffer & put** (int index, double value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )

*Writes the given doubles into this buffer at the given index.*

- virtual **DoubleBuffer \* slice** () const =0

*Creates a new **DoubleBuffer** (p. 1865) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **DoubleBuffer** &value) const

- virtual bool **equals** (const **DoubleBuffer** &value) const

- virtual bool **operator==** (const **DoubleBuffer** &value) const
- virtual bool **operator<** (const **DoubleBuffer** &value) const

### Static Public Member Functions

- static **DoubleBuffer** \* **allocate** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Allocates a new **DoubleBuffer** (p. 1865).*
- static **DoubleBuffer** \* **wrap** (double \*array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Wraps the passed buffer with a new **DoubleBuffer** (p. 1865).*
- static **DoubleBuffer** \* **wrap** (std::vector< double > &buffer)  
*Wraps the passed STL double Vector in a **DoubleBuffer** (p. 1865).*

### Protected Member Functions

- **DoubleBuffer** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **DoubleBuffer** (p. 1865) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

#### 6.339.1 Detailed Description

This class defines four categories of operations upon double buffers: o Absolute and relative get and put methods that read and write single doubles; o Relative bulk get methods that transfer contiguous sequences of doubles from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of doubles from a double array or some other double buffer into this buffer o Methods for compacting, duplicating, and slicing a double buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing double array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

**Since**

1.0

## 6.339.2 Constructor & Destructor Documentation

**6.339.2.1** `decaf::nio::DoubleBuffer::DoubleBuffer ( int capacity ) throw ( decaf::lang::exceptions::IllegalArgumentException )` [protected]

Creates a **DoubleBuffer** (p. 1865) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

### Parameters

<i>capacity</i>	The size and limit of the <b>Buffer</b> (p. 936) in doubles
-----------------	---

### Exceptions

<i>IllegalArgumentException</i>	if capacity is negative.
---------------------------------	--------------------------

**6.339.2.2** `virtual decaf::nio::DoubleBuffer::~~DoubleBuffer ( )` [inline, virtual]

## 6.339.3 Member Function Documentation

**6.339.3.1** `static DoubleBuffer* decaf::nio::DoubleBuffer::allocate ( int capacity ) throw ( decaf::lang::exceptions::IllegalArgumentException )` [static]

Allocates a new **DoubleBuffer** (p. 1865).

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

### Parameters

<i>capacity</i>	The size of the Double buffer in doubles.
-----------------	---

### Returns

the **DoubleBuffer** (p. 1865) that was allocated, caller owns.

### Exceptions

<i>IllegalArgumentException</i>	is the capacity value is negative.
---------------------------------	------------------------------------

**6.339.3.2** `virtual double* decaf::nio::DoubleBuffer::array ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )` [pure virtual]

Returns the double array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

the array that backs this **Buffer** (p. 936).

### Exceptions

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<i><b>UnsupportedOperationException</b></i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1859).

**6.339.3.3** `virtual int decaf::nio::DoubleBuffer::arrayOffset ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )` [pure virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

The offset into the backing array where index zero starts.

### Exceptions

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<i><b>UnsupportedOperationException</b></i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1860).

**6.339.3.4** `virtual DoubleBuffer* decaf::nio::DoubleBuffer::asReadOnlyBuffer ( ) const` [pure virtual]

Creates a new, read-only double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and

will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

### Returns

The new, read-only double buffer which the caller then owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1860).

**6.339.3.5** `virtual DoubleBuffer& decaf::nio::DoubleBuffer::compact ( ) throw ( ReadOnlyBufferException ) [pure virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 941) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 940) - 1 is copied to index  $n = \text{limit}()$  (p. 940) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

### Returns

a reference to this **DoubleBuffer** (p. 1865).

### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.
--	------------------------------

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1861).

**6.339.3.6** `virtual int decaf::nio::DoubleBuffer::compareTo ( const DoubleBuffer & value ) const [virtual]`

**6.339.3.7** `virtual DoubleBuffer* decaf::nio::DoubleBuffer::duplicate ( ) [pure virtual]`

Creates a new double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content



will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

a new double **Buffer** (p. 936) which the caller owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1861).

**6.339.3.8** `virtual bool decaf::nio::DoubleBuffer::equals ( const DoubleBuffer & value ) const`  
[virtual]

**6.339.3.9** `DoubleBuffer& decaf::nio::DoubleBuffer::get ( std::vector< double > buffer )`  
`throw ( BufferUnderflowException )`

Relative bulk get method.

This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize( N )` before calling this get method.

### Returns

a reference to this **Buffer** (p. 936).

### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 967)	if there are fewer than length doubles remaining in this buffer
--	---

**6.339.3.10** `virtual double decaf::nio::DoubleBuffer::get ( int index ) const throw (`  
`lang::exceptions::IndexOutOfBoundsException )` [pure virtual]

Absolute get method.

Reads the value at the given index.

### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the double is to be read.
--------------	---

### Returns

the double that is located at the given index.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit
----------------------------------	---

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1862).

**6.339.3.11 DoubleBuffer& decaf::nio::DoubleBuffer::get ( double \* *buffer*, int *size*, int *offset*, int *length* ) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )**

Relative bulk get method.

This method transfers doubles from this buffer into the given destination array. If there are fewer doubles remaining in the buffer than are required to satisfy the request, that is, if *length* > **remaining()** (p. 941), then no bytes are transferred and a **BufferUnderflowException** (p. 967) is thrown.

Otherwise, this method copies *length* doubles from this buffer into the given array, starting at the current position of this buffer and at the given *offset* in the array. The position of this buffer is then incremented by *length*.

**Parameters**

<i>buffer</i>	The pointer to an allocated buffer to fill.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The amount of data to put in the passed buffer.

**Returns**

a reference to this **Buffer** (p. 936).

**Exceptions**

<b>BufferUnderflowException</b> (p. 967)	if there are fewer than <i>length</i> doubles remaining in this buffer
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of <i>size</i> , <i>offset</i> , or <i>length</i> are not met.

**6.339.3.12 virtual double decaf::nio::DoubleBuffer::get ( ) throw ( BufferUnderflowException ) [pure virtual]**

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

**Returns**

the double at the current position.

**Exceptions**

<i><b>BufferUnderflowException</b></i> (p. 967)	if there no more data to return.
--	----------------------------------

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1862).

**6.339.3.13** `virtual bool decaf::nio::DoubleBuffer::hasArray ( ) const` [pure virtual]

Tells whether or not this buffer is backed by an accessible double array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns**

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1863).

**6.339.3.14** `virtual bool decaf::nio::DoubleBuffer::operator< ( const DoubleBuffer & value ) const` [virtual]

**6.339.3.15** `virtual bool decaf::nio::DoubleBuffer::operator== ( const DoubleBuffer & value ) const` [virtual]

**6.339.3.16** `virtual DoubleBuffer& decaf::nio::DoubleBuffer::put ( double value ) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes the given doubles into this buffer at the current position, and then increments the position.

**Parameters**

<i>value</i>	The doubles value to be written.
--------------	----------------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<i><b>BufferOverflowException</b></i> (p. 964)	if this buffer's current position is not smaller than its limit.
---	--

<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.
--	------------------------------

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1864).

**6.339.3.17** **virtual DoubleBuffer& decaf::nio::DoubleBuffer::put ( int *index*, double *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )** [pure virtual]

Writes the given doubles into this buffer at the given index.

#### Parameters

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data.
<i>value</i>	The doubles to write.

#### Returns

a reference to this buffer

#### Exceptions

<b><i>IndexOutOfBoundsException</i></b>	if index greater than the buffer's limit minus the size of the type being written, or the index is negative.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1863).

**6.339.3.18** **DoubleBuffer& decaf::nio::DoubleBuffer::put ( const double \* *buffer*, int *size*, int *offset*, int *length* ) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )**

This method transfers doubles into this buffer from the given source array.

If there are more doubles to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 941), then no doubles are transferred and a **BufferOverflowException** (p. 964) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given `offset` in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

#### Parameters

<i>buffer</i>	The array from which doubles are to be read.
---------------	--

<i>size</i>	The size of the buffer passed.
<i>offset</i>	The offset within the array of the first char to be read.
<i>length</i>	The number of doubles to be read from the given array.

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 964)	if there is insufficient space in this buffer
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only
<b><i>NullPointerException</i></b>	if the passed buffer is null.
<b><i>IndexOutOfBoundsException</i></b>	if the preconditions of size, offset, or length are not met.

### 6.339.3.19 DoubleBuffer& decaf::nio::DoubleBuffer::put ( std::vector< double > & buffer ) throw ( BufferOverflowException, ReadOnlyBufferException )

This method transfers the entire content of the given source doubles array into this buffer.

This is the same as calling put( &buffer[0], 0, buffer.size()).

**Parameters**

<i>buffer</i>	The buffer whose contents are copied to this <b>DoubleBuffer</b> (p. 1865).
---------------	---

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 964)	if there is insufficient space in this buffer.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.

**6.339.3.20** `DoubleBuffer& decaf::nio::DoubleBuffer::put ( DoubleBuffer & src )`  
`throw ( BufferOverflowException, ReadOnlyBufferException,`  
`decaf::lang::exceptions::IllegalArgumentException )`

This method transfers the doubles remaining in the given source buffer into this buffer.

If there are more doubles remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 941), then no doubles are transferred and a **BufferOverflowException** (p. 964) is thrown.

Otherwise, this method copies `n = src.remaining()` doubles from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

#### Parameters

<i>src</i>	The buffer to take doubles from an place in this one.
------------	---

#### Returns

a reference to this buffer.

#### Exceptions

<b><i>BufferOverflowException</i></b> (p. 964)	if there is insufficient space in this buffer for the remaining doubles in the source buffer
<b><i>IllegalArgumentException</i></b> <i>ception</i>	if the source buffer is this buffer.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.

**6.339.3.21** `virtual DoubleBuffer* decaf::nio::DoubleBuffer::slice ( ) const` [pure  
virtual]

Creates a new **DoubleBuffer** (p. 1865) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the newly create **DoubleBuffer** (p. 1865) which the caller owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1864).

**6.339.3.22** `virtual std::string decaf::nio::DoubleBuffer::toString ( ) const` `[virtual]`

#### Returns

a std::string describing this object

**6.339.3.23** `static DoubleBuffer* decaf::nio::DoubleBuffer::wrap ( double * array, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` `[static]`

Wraps the passed buffer with a new **DoubleBuffer** (p. 1865).

The new buffer will be backed by the given double array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>array</i>	The array that will back the new buffer.
<i>size</i>	The size of the passed in array.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

#### Returns

a new **DoubleBuffer** (p. 1865) that is backed by buffer, caller owns.

#### Exceptions

<i>NullPointerException</i>	if the array pointer is NULL.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

**6.339.3.24** `static DoubleBuffer* decaf::nio::DoubleBuffer::wrap ( std::vector< double > & buffer )` `[static]`

Wraps the passed STL double Vector in a **DoubleBuffer** (p. 1865).

The new buffer will be backed by the given double array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).
---------------	--

**Returns**

a new **DoubleBuffer** (p. 1865) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**DoubleBuffer.h**

**6.340 decaf::lang::DYNAMIC\_CAST\_TOKEN Struct Reference**

```
#include <src/main/decaf/lang/Pointer.h>
```

The documentation for this struct was generated from the following file:

- src/main/decaf/lang/**Pointer.h**

**6.341 activemq::cmsutil::DynamicDestinationResolver Class Reference**

Resolves a CMS destination name to a *Destination*.

```
#include <src/main/activemq/cmsutil/DynamicDestinationResolver.h>
```

Inheritance diagram for activemq::cmsutil::DynamicDestinationResolver:

**Data Structures**

- class **SessionResolver**  
*Manages maps of names to topics and queues for a single session.*

**Public Member Functions**

- **DynamicDestinationResolver** ()
- virtual **~DynamicDestinationResolver** ()
- virtual void **init** (**ResourceLifecycleManager** \*mgr)  
*Initializes this destination resolver for use.*
- virtual void **destroy** ()  
*Destroys any allocated resources.*
- virtual **cms::Destination** \* **resolveDestinationName** (**cms::Session** \*session, const std::string &destName, bool pubSubDomain) throw ( cms::CMSException )



*Resolves the given name to a destination.*

### Protected Member Functions

- **DynamicDestinationResolver** (const **DynamicDestinationResolver** &)
- **DynamicDestinationResolver** & **operator=** (const **DynamicDestinationResolver** &)

#### 6.341.1 Detailed Description

Resolves a CMS destination name to a `Destination`.

#### 6.341.2 Constructor & Destructor Documentation

6.341.2.1 **activemq::cmsutil::DynamicDestinationResolver::DynamicDestinationResolver** ( const **DynamicDestinationResolver** & ) [inline, protected]

6.341.2.2 **activemq::cmsutil::DynamicDestinationResolver::DynamicDestinationResolver** ( )

6.341.2.3 **virtual activemq::cmsutil::DynamicDestinationResolver::~~DynamicDestinationResolver** ( ) [virtual]

#### 6.341.3 Member Function Documentation

6.341.3.1 **virtual void activemq::cmsutil::DynamicDestinationResolver::destroy** ( ) [virtual]

Destroys any allocated resources.

Implements **activemq::cmsutil::DestinationResolver** (p. 1811).

6.341.3.2 **virtual void activemq::cmsutil::DynamicDestinationResolver::init** ( **ResourceLifecycleManager** \* *mgr* ) [inline, virtual]

Initializes this destination resolver for use.

Ensures that any previously allocated resources are first destroyed (e.g. calls **destroy()** (p. 1879)).

#### Parameters

<i>mgr</i>	the resource lifecycle manager.
------------	---------------------------------

Implements **activemq::cmsutil::DestinationResolver** (p. 1811).

- 6.341.3.3 **DynamicDestinationResolver**& **activemq::cmsutil::DynamicDestinationResolver::operator=** ( **const DynamicDestinationResolver &** ) [**inline**, **protected**]
- 6.341.3.4 **virtual cms::Destination\*** **activemq::cmsutil::DynamicDestinationResolver::resolveDestinationName** ( **cms::Session \*** *session*, **const std::string &** *destName*, **bool** *pubSubDomain* ) **throw** ( **cms::CMSException** ) [**virtual**]

Resolves the given name to a destination.

If *pubSubDomain* is true, a topic will be returned, otherwise a queue will be returned.

#### Parameters

<i>session</i>	the session for which to retrieve resolve the destination.
<i>destName</i>	the name to be resolved.
<i>pubSubDomain</i>	If true, the name will be resolved to a Topic, otherwise a Queue.

#### Returns

the resolved destination

#### Exceptions

<b>cms::CMSException</b> (p. 1190)	if resolution failed.
---------------------------------------	-----------------------

Implements **activemq::cmsutil::DestinationResolver** (p. 1811).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/DynamicDestinationResolver.h`

## 6.342 decaf::util::Map< K, V, COMPARATOR >::Entry Class Reference

```
#include <src/main/decaf/util/Map.h>
```

#### Public Member Functions

- **Entry** ()
- **virtual ~Entry** ()
- **virtual const K & getKey** () **const** =0
- **virtual const V & getValue** () **const** =0
- **virtual void setValue** (const V &value)=0

```
template<typename K, typename V, typename COMPARATOR = std::less<K>> class decaf::util::Map<
K, V, COMPARATOR >::Entry
```

### 6.342.1 Constructor & Destructor Documentation

6.342.1.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::Map< K, V, COMPARATOR >::Entry::Entry ( ) [inline]`

6.342.1.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual decaf::util::Map< K, V, COMPARATOR >::Entry::~Entry ( ) [inline, virtual]`

### 6.342.2 Member Function Documentation

6.342.2.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual const K& decaf::util::Map< K, V, COMPARATOR >::Entry::getKey ( ) const [pure virtual]`

6.342.2.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual const V& decaf::util::Map< K, V, COMPARATOR >::Entry::getValue ( ) const [pure virtual]`

6.342.2.3 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR >::Entry::setValue ( const V & value ) [pure virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Map.h`

## 6.343 decaf::io::EOFException Class Reference

```
#include <src/main/decaf/io/EOFException.h>
```

Inheritance diagram for decaf::io::EOFException:

### Public Member Functions

- **EOFException** () throw ()  
*Default Constructor.*
- **EOFException** (const **lang::Exception** &ex) throw ()  
*Copy Constructor.*
- **EOFException** (const **EOFException** &ex) throw ()

*Copy Constructor.*

- **EOFException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()

*Constructor - Initializes the file name and line number where this message occurred.*

- **EOFException** (const std::exception \*cause) throw ()

*Constructor.*

- **EOFException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()

*Constructor.*

- virtual **EOFException** \* clone () const

*Clones this exception.*

- virtual ~**EOFException** () throw ()

### 6.343.1 Constructor & Destructor Documentation

#### 6.343.1.1 decaf::io::EOFException::EOFException ( ) throw () [inline]

Default Constructor.

#### 6.343.1.2 decaf::io::EOFException::EOFException ( const lang::Exception & ex ) throw () [inline]

Copy Constructor.

##### Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

#### 6.343.1.3 decaf::io::EOFException::EOFException ( const EOFException & ex ) throw () [inline]

Copy Constructor.

##### Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

**6.343.1.4** `decaf::io::EOFException::EOFException ( const char * file, const int lineNumber,  
const std::exception * cause, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.343.1.5** `decaf::io::EOFException::EOFException ( const std::exception * cause ) throw ()`  
`[inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.343.1.6** `decaf::io::EOFException::EOFException ( const char * file, const int lineNumber,  
const char * msg, ... ) throw ()` `[inline]`

Constructor.

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.343.1.7** `virtual decaf::io::EOFException::~EOFException ( ) throw ()` `[inline, virtual]`

## 6.343.2 Member Function Documentation

**6.343.2.1** `virtual EOFException* decaf::io::EOFException::clone ( ) const` `[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

a new instance of an Exception that is a copy of this one.

Reimplemented from **decaf::io::IOException** (p. 2212).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/EOFException.h`

## 6.344 decaf::util::logging::ErrorManager Class Reference

**ErrorManager** (p. 1884) objects can be attached to Handlers to process any error that occur on a **Handler** (p. 2042) during Logging.

```
#include <src/main/decaf/util/logging/ErrorManager.h>
```

### Public Member Functions

- **ErrorManager** ()
- virtual **~ErrorMessage** ()
- virtual void **error** (const std::string &message, **decaf::lang::Exception** \*ex, int code)

*The error method is called when a **Handler** (p. 2042) failure occurs.*

### Static Public Attributes

- static const int **GENERIC\_FAILURE**  
*GENERIC\_FAILURE is used for failure that don't fit into one of the other categories.*
- static const int **WRITE\_FAILURE**  
*WRITE\_FAILURE is used when a write to an output stream fails.*
- static const int **FLUSH\_FAILURE**  
*FLUSH\_FAILURE is used when a flush to an output stream fails.*
- static const int **CLOSE\_FAILURE**  
*CLOSE\_FAILURE is used when a close of an output stream fails.*
- static const int **OPEN\_FAILURE**  
*OPEN\_FAILURE is used when an open of an output stream fails.*

- static const int **FORMAT\_FAILURE**

*FORMAT\_FAILURE is used when formatting fails for any reason.*

### 6.344.1 Detailed Description

**ErrorManager** (p. 1884) objects can be attached to Handlers to process any error that occur on a **Handler** (p. 2042) during Logging. When processing logging output, if a **Handler** (p. 2042) encounters problems then rather than throwing an Exception back to the issuer of the logging call (who is unlikely to be interested) the **Handler** (p. 2042) should call its associated **ErrorManager** (p. 1884).

Since

1.0

### 6.344.2 Constructor & Destructor Documentation

6.344.2.1 `decaf::util::logging::ErrorManager::ErrorManager ( )`

6.344.2.2 `virtual decaf::util::logging::ErrorManager::~~ErrorManager ( )` [virtual]

### 6.344.3 Member Function Documentation

6.344.3.1 `virtual void decaf::util::logging::ErrorManager::error ( const std::string & message, decaf::lang::Exception * ex, int code )` [virtual]

The error method is called when a **Handler** (p. 2042) failure occurs.

This method may be overridden in subclasses. The default behavior in this base class is that the first call is reported to System.err, and subsequent calls are ignored.

#### Parameters

<i>msg</i>	- a descriptive string (may be empty)
<i>ex</i>	- an exception (may be NULL)
<i>code</i>	- an error code defined in <b>ErrorManager</b> (p. 1884)

### 6.344.4 Field Documentation

6.344.4.1 `const int decaf::util::logging::ErrorManager::CLOSE_FAILURE`  
[static]

CLOSE\_FAILURE is used when a close of an output stream fails.

**6.344.4.2** `const int decaf::util::logging::ErrorManager::FLUSH_FAILURE`  
[static]

FLUSH\_FAILURE is used when a flush to an output stream fails.

**6.344.4.3** `const int decaf::util::logging::ErrorManager::FORMAT_FAILURE`  
[static]

FORMAT\_FAILURE is used when formatting fails for any reason.

**6.344.4.4** `const int decaf::util::logging::ErrorManager::GENERIC_FAILURE`  
[static]

GENERIC\_FAILURE is used for failure that don't fit into one of the other categories.

**6.344.4.5** `const int decaf::util::logging::ErrorManager::OPEN_FAILURE`  
[static]

OPEN\_FAILURE is used when an open of an output stream fails.

**6.344.4.6** `const int decaf::util::logging::ErrorManager::WRITE_FAILURE`  
[static]

WRITE\_FAILURE is used when a write to an output stream fails.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/ErrorManager.h`

## 6.345 decaf::lang::Exception Class Reference

```
#include <src/main/decaf/lang/Exception.h>
```

Inheritance diagram for decaf::lang::Exception:

### Public Member Functions

- **Exception** () throw ()  
*Default Constructor.*
- **Exception** (const **Exception** &ex) throw ()  
*Copy Constructor.*



- **Exception** (const std::exception \***cause**) throw ()  
*Constructor.*
- **Exception** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **Exception** (const char \*file, const int lineNumber, const std::exception \***cause**, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual ~**Exception** () throw ()
- virtual std::string **getMessage** () const  
*Gets the message for this exception.*
- virtual const std::exception \* **getCause** () const  
*Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.*
- virtual void **initCause** (const std::exception \***cause**)  
*Initializes the contained cause exception with the one given.*
- virtual const char \* **what** () const throw ()  
*Implement method from std::exception.*
- virtual void **setMessage** (const char \*msg,...)  
*Sets the cause for this exception.*
- virtual void **setMark** (const char \*file, const int lineNumber)  
*Adds a file/line number to the stack trace.*
- virtual **Exception** \* **clone** () const  
*Clones this exception.*
- virtual std::vector< std::pair< std::string, int > > **getStackTrace** () const  
*Provides the stack trace for every point where this exception was caught, marked, and rethrown.*
- virtual void **printStackTrace** () const  
*Prints the stack trace to std::err.*
- virtual void **printStackTrace** (std::ostream &stream) const  
*Prints the stack trace to the given output stream.*
- virtual std::string **getStackTraceString** () const

*Gets the stack trace as one contiguous string.*

- **Exception & operator=** (const **Exception** &ex)

*Assignment operator.*

## Protected Member Functions

- virtual void **setStackTrace** (const std::vector< std::pair< std::string, int > > &trace)
- virtual void **buildMessage** (const char \*format, va\_list &vargs)

## Protected Attributes

- std::string **message**  
*The cause of this exception.*
- std::exception \* **cause**  
*The **Exception** (p. 1886) that caused this one to be thrown.*
- std::vector< std::pair< std::string, int > > **stackTrace**  
*The stack trace.*

## 6.345.1 Constructor & Destructor Documentation

### 6.345.1.1 decaf::lang::Exception::Exception ( ) throw ()

Default Constructor.

### 6.345.1.2 decaf::lang::Exception::Exception ( const **Exception** & ex ) throw ()

Copy Constructor.

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) instance to copy.
-----------	--

### 6.345.1.3 decaf::lang::Exception::Exception ( const std::exception \* *cause* ) throw ()

Constructor.

#### Parameters

<i>cause</i>	<b>Pointer</b> (p. 3032) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

**6.345.1.4** `decaf::lang::Exception::Exception ( const char * file, const int lineNumber, const char * msg, ... ) throw ()`

Constructor - Initializes the file name and line number where this message occurred.  
Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.345.1.5** `decaf::lang::Exception::Exception ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()`

Constructor - Initializes the file name and line number where this message occurred.  
Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.345.1.6** `virtual decaf::lang::Exception::~~Exception ( ) throw ()` [virtual]

### 6.345.2 Member Function Documentation

**6.345.2.1** `virtual void decaf::lang::Exception::buildMessage ( const char * format, va_list & vargs )` [protected, virtual]

Referenced by decaf::lang::exceptions::NumberFormatException::NumberFormatException().

**6.345.2.2** `virtual Exception* decaf::lang::Exception::clone ( ) const` [virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

Copy of this **Exception** (p. 1886) object

Implements **decaf::lang::Throwable** (p. 3896).

Reimplemented in **activemq::exceptions::ActiveMQException** (p. 349), **activemq::exceptions::BrokerException** (p. 874), **decaf::internal::net::ssl::openssl::OpenSSLSocketException** (p. 2958), **decaf::io::EOFException** (p. 1883), **decaf::io::InterruptedIOException** (p. 2198), **decaf::io::IOException** (p. 2212), **decaf::io::UnsupportedEncodingException** (p. 4027), **decaf::io::UTFDataFormatException** (p. 4080), **decaf::lang::exceptions::ClassCastException** (p. 1179), **decaf::lang::exceptions::IllegalArgumentException** (p. 2056), **decaf::lang::exceptions::IllegalMonitorStateException** (p. 2059), **decaf::lang::exceptions::IllegalStateException** (p. 2062), **decaf::lang::exceptions::IllegalThreadStateException** (p. 2065), **decaf::lang::exceptions::IndexOutOfBoundsException** (p. 2072), **decaf::lang::exceptions::InterruptedException** (p. 2196), **decaf::lang::exceptions::InvalidStateException** (p. 2210), **decaf::lang::exceptions::NoSuchElementException** (p. 2912), **decaf::lang::exceptions::NullPointerException** (p. 2918), **decaf::lang::exceptions::NumberFormatException** (p. 2923), **decaf::lang::exceptions::RuntimeException** (p. 3423), **decaf::lang::exceptions::UnsupportedOperationException** (p. 4030), **decaf::net::BindException** (p. 844), **decaf::net::ConnectException** (p. 1296), **decaf::net::HttpRetryException** (p. 2051), **decaf::net::MalformedURLException** (p. 2538), **decaf::net::NoRouteToHostException** (p. 2907), **decaf::net::PortUnreachableException** (p. 3062), **decaf::net::ProtocolException** (p. 3230), **decaf::net::SocketException** (p. 3629), **decaf::net::SocketTimeoutException** (p. 3652), **decaf::net::UnknownHostException** (p. 4022), **decaf::net::UnknownServiceException** (p. 4024), **decaf::net::URISyntaxException** (p. 4063), **decaf::nio::BufferOverflowException** (p. 966), **decaf::nio::BufferUnderflowException** (p. 969), **decaf::nio::InvalidMarkException** (p. 2206), **decaf::nio::ReadOnlyBufferException** (p. 3263), **decaf::security::cert::CertificateEncodingException** (p. 1118), **decaf::security::cert::CertificateExpiredException** (p. 1120), **decaf::security::cert::CertificateExpiredException** (p. 1122), **decaf::security::cert::CertificateNotFoundException** (p. 1124), **decaf::security::cert::CertificateParsingException** (p. 1126), **decaf::security::GeneralSecurityException** (p. 2037), **decaf::security::InvalidKeyException** (p. 2203), **decaf::security::KeyException** (p. 2368), **decaf::security::KeyManagementException** (p. 2371), **decaf::security::NoSuchAlgorithmException** (p. 2910), **decaf::security::NoSuchProviderException** (p. 2915), **decaf::security::SignatureException** (p. 3604), **decaf::util::concurrent::BrokenBarrierException** (p. 868), **decaf::util::concurrent::CancellationException** (p. 1112), **decaf::util::concurrent::ExecutionException** (p. 1926), **decaf::util::concurrent::RejectedExecutionException** (p. 3282), **decaf::util::concurrent::TimeoutException** (p. 3901), **decaf::util::zip::DataFormatException** (p. 1602), and **decaf::util::zip::ZipException** (p. 4178).

```
6.345.2.3 virtual const std::exception* decaf::lang::Exception::getCause ( ) const
[inline, virtual]
```

Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

### Returns

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

Implements **decaf::lang::Throwable** (p. 3897).

**6.345.2.4** `virtual std::string decaf::lang::Exception::getMessage ( ) const [inline, virtual]`

Gets the message for this exception.

#### Returns

Text formatted error message

Implements **decaf::lang::Throwable** (p. 3897).

Referenced by `activemq::exceptions::BrokerException::BrokerException()`.

**6.345.2.5** `virtual std::vector< std::pair< std::string, int> > decaf::lang::Exception::getStackTrace ( ) const [virtual]`

Provides the stack trace for every point where this exception was caught, marked, and rethrown.

The first item in the returned vector is the first point where the mark was set (e.g. where the exception was created).

#### Returns

the stack trace.

Implements **decaf::lang::Throwable** (p. 3897).

**6.345.2.6** `virtual std::string decaf::lang::Exception::getStackTraceString ( ) const [virtual]`

Gets the stack trace as one contiguous string.

#### Returns

string with formatted stack trace data

Implements **decaf::lang::Throwable** (p. 3898).

**6.345.2.7** `virtual void decaf::lang::Exception::initCause ( const std::exception * cause ) [virtual]`

Initializes the contained cause exception with the one given.

A copy is made to avoid ownership issues.

#### Parameters

<i>cause</i>	The exception that was the cause of this one.
--------------	---

Implements **decaf::lang::Throwable** (p. 3898).

#### 6.345.2.8 **Exception& decaf::lang::Exception::operator= ( const Exception & ex )**

Assignment operator.

##### Parameters

<i>ex</i>	const reference to another <b>Exception</b> (p. 1886)
-----------	---

#### 6.345.2.9 **virtual void decaf::lang::Exception::printStackTrace ( ) const** [virtual]

Prints the stack trace to std::err.

Implements **decaf::lang::Throwable** (p. 3898).

#### 6.345.2.10 **virtual void decaf::lang::Exception::printStackTrace ( std::ostream & stream ) const** [virtual]

Prints the stack trace to the given output stream.

##### Parameters

<i>stream</i>	the target output stream.
---------------	---------------------------

Implements **decaf::lang::Throwable** (p. 3898).

#### 6.345.2.11 **virtual void decaf::lang::Exception::setMark ( const char \* file, const int lineNumber )** [virtual]

Adds a file/line number to the stack trace.

##### Parameters

<i>file</i>	The name of the file calling this method (use __FILE__).
<i>lineNumber</i>	The line number in the calling file (use __LINE__).

Implements **decaf::lang::Throwable** (p. 3899).

Referenced by decaf::lang::exceptions::NumberFormatException::NumberFormatException().

#### 6.345.2.12 **virtual void decaf::lang::Exception::setMessage ( const char \* msg, ... )** [virtual]

Sets the cause for this exception.

**Parameters**

<i>msg</i>	the format string for the msg.
...	params to format into the string

**6.345.2.13** `virtual void decaf::lang::Exception::setStackTrace ( const std::vector< std::pair< std::string, int > > & trace )` [protected, virtual]

**6.345.2.14** `virtual const char* decaf::lang::Exception::what ( ) const throw ()` [inline, virtual]

Implement method from std::exception.

**Returns**

the const char\* of `getMessage ()` (p. 1891).

**6.345.3 Field Documentation**

**6.345.3.1** `std::exception* decaf::lang::Exception::cause` [protected]

The **Exception** (p. 1886) that caused this one to be thrown.

**6.345.3.2** `std::string decaf::lang::Exception::message` [protected]

The cause of this exception.

**6.345.3.3** `std::vector< std::pair< std::string, int> > decaf::lang::Exception::stackTrace` [protected]

The stack trace.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Exception.h**

**6.346 cms::ExceptionListener Class Reference**

If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 1893) that is registered with the **Connection** (p. 1296).

```
#include <src/main/cms/ExceptionListener.h>
```

**Public Member Functions**

- virtual `~ExceptionListener ()`

- virtual void **onException** (const **cms::CMSException** &ex)=0

*Called when an exception occurs.*

### 6.346.1 Detailed Description

If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 1893) that is registered with the **Connection** (p. 1296). An exception listener allows a client to be notified of a problem asynchronously. Some connections only consume messages via the asynchronous event mechanism so they would have no other way to learn that their connection has failed.

Since

1.0

### 6.346.2 Constructor & Destructor Documentation

- 6.346.2.1 virtual **cms::ExceptionListener::~ExceptionListener** ( ) [inline, virtual]

### 6.346.3 Member Function Documentation

- 6.346.3.1 virtual void **cms::ExceptionListener::onException** ( const **cms::CMSException** &ex ) [pure virtual]

Called when an exception occurs.

Once notified of an exception the caller should no longer use the resource that generated the exception.

#### Parameters

<i>ex</i>	Exception Object that occurred.
-----------	---------------------------------

The documentation for this class was generated from the following file:

- src/main/cms/**ExceptionListener.h**

## 6.347 activemq::commands::ExceptionResponse Class Reference

```
#include <src/main/activemq/commands/ExceptionResponse.h>
```

Inheritance diagram for **activemq::commands::ExceptionResponse**:



## Public Member Functions

- **ExceptionResponse** ()
- virtual **~ExceptionResponse** ()
- virtual unsigned char **getDataStructureType** () const

*Get the unique identifier that this object and its own Marshaler share.*

- virtual **ExceptionResponse \* cloneDataStructure** () const

*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*

- virtual void **copyDataStructure** (const **DataStructure** \*src)

*Copy the contents of the passed object into this object's members, overwriting any existing data.*

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** \*value) const

*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **BrokerError** > & **getException** () const
- virtual **Pointer**< **BrokerError** > & **getException** ()
- virtual void **setException** (const **Pointer**< **BrokerError** > &exception)

## Static Public Attributes

- static const unsigned char **ID\_EXCEPTIONRESPONSE** = 31

## Protected Attributes

- **Pointer**< **BrokerError** > **exception**

### 6.347.1 Constructor & Destructor Documentation

6.347.1.1 `activemq::commands::ExceptionResponse::ExceptionResponse ( )`

6.347.1.2 `virtual activemq::commands::ExceptionResponse::~~ExceptionResponse ( )`  
[virtual]

### 6.347.2 Member Function Documentation

6.347.2.1 `virtual ExceptionResponse* activemq::commands::ExceptionResponse::cloneDataStructure ( ) const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Reimplemented from `activemq::commands::Response` (p. 3380).

6.347.2.2 `virtual void activemq::commands::ExceptionResponse::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from `activemq::commands::Response` (p. 3380).

6.347.2.3 `virtual bool activemq::commands::ExceptionResponse::equals ( const DataStructure * value ) const` [virtual]

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::Response` (p. 3381).

6.347.2.4 `virtual unsigned char activemq::commands::ExceptionResponse::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataSet** (p. 1713) type copy.

Reimplemented from **activemq::commands::Response** (p. 3381).

6.347.2.5 `virtual Pointer<BrokerError>& activemq::commands::ExceptionResponse::getException ( ) [virtual]`

6.347.2.6 `virtual const Pointer<BrokerError>& activemq::commands::ExceptionResponse::getException ( ) const [virtual]`

6.347.2.7 `virtual void activemq::commands::ExceptionResponse::setException ( const Pointer< BrokerError > & exception ) [virtual]`

6.347.2.8 `virtual std::string activemq::commands::ExceptionResponse::toString ( ) const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 3382).

### 6.347.3 Field Documentation

6.347.3.1 `Pointer<BrokerError> activemq::commands::ExceptionResponse::exception [protected]`

6.347.3.2 `const unsigned char activemq::commands::ExceptionResponse::ID_EXCEPTIONRESPONSE = 31 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ExceptionResponse.h`

## 6.348 activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1898).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ExceptionResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller:

### Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

6.348

activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller  
Class Reference 1901

### 6.348.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1898).  
NOTE!: This file is auto generated - do not modify! if you need to make a change,  
please see the Java Classes in the activemq-openwire-generator module

### 6.348.2 Constructor & Destructor Documentation

6.348.2.1 **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::ExceptionResponseMarshaller**  
( ) [inline]

6.348.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::~~ExceptionResponseMarshaller**  
( ) [inline, virtual]

### 6.348.3 Member Function Documentation

6.348.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller**  
(p. 3415).

6.348.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller**  
(p. 3415).

6.348.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3415).

**6.348.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3416).

**6.348.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

6.348

**activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller****Class Reference****1903****Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3417).

**6.348.3.6** virtual void **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::tightMarshal2**  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
 \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*,  
**utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** )  
 [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3417).

**6.348.3.7** virtual void **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::tightUnmarshal**  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream**  
 \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3418).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ExceptionResponseMarshaller.h`

## 6.349 **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1902).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ExceptionResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller**:

**Public Member Functions**

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*



## 6.349

**activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller**

### Class Reference

1905

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.349.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1902).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.349.2 Constructor & Destructor Documentation

6.349.2.1 **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::ExceptionResponseMarshaller**  
( ) [inline]

6.349.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::~~ExceptionResponseMarshaller**  
( ) [inline, virtual]

### 6.349.3 Member Function Documentation

6.349.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3395).

6.349.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3395).

**6.349.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3395).

**6.349.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3396).

6.349

activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller

Class Reference

1907

```
6.349.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3397).

```
6.349.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3397).

```
6.349.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3398).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ExceptionResponseMarshaller.h

## 6.350 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1906).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller**:

#### Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

## 6.350

**activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller**

**Class Reference**

**1909**

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.350.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1906).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.350.2 Constructor & Destructor Documentation

6.350.2.1 **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::ExceptionResponseMarshaller**  
( ) [inline]

6.350.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::~~ExceptionResponseMarshaller**  
( ) [inline, virtual]

### 6.350.3 Member Function Documentation

6.350.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3405).

**6.350.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3405).

**6.350.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3405).

**6.350.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

6.350

**activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller**

**Class Reference**

**1911**

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3406).

```
6.350.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::tightMarshal1  
( OpenWireFormat * wireFormat, commands::DataStructure *  
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )  
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3407).

```
6.350.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::tightMarshal2  
( OpenWireFormat * wireFormat, commands::DataStructure  
  * dataStructure, decaf::io::DataOutputStream * dataOut,  
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )  
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3407).

**6.350.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3408).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h`

## 6.351 **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1910).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller**:



## Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.351.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1910).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.351.2 Constructor & Destructor Documentation

6.351.2.1 `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::ExceptionResponseMarshaller ( ) [inline]`

6.351.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::~~ExceptionResponseMarshaller ( ) [inline, virtual]`

### 6.351.3 Member Function Documentation

6.351.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3400).

6.351.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3400).

6.351.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

6.351

**activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller**  
**Class Reference** **1915**  
**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3400).

**6.351.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3401).

**6.351.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3402).

```
6.351.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3402).

```
6.351.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3403).

The documentation for this class was generated from the following file:

6.352

**activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller**

**Class Reference**

1917

- `src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h`

## 6.352 **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** **Class Reference**

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1915).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller`:

### Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.352.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1915).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.352.2 Constructor & Destructor Documentation

6.352.2.1 **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::ExceptionResponseMarshaller**  
 ( ) [inline]

6.352.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::~~ExceptionResponseMarshaller**  
 ( ) [inline, virtual]

### 6.352.3 Member Function Documentation

6.352.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller**  
 (p. 3390).

6.352.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller**  
 (p. 3390).

6.352

activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller

Class Reference

1919

6.352.3.3 virtual void activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::looseMarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure  
\* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw (  
decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller**  
(p. 3390).

6.352.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::looseUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure  
\* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw (  
decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller**  
(p. 3391).

6.352.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::tightMarshal1  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3392).

```
6.352.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3392).

```
6.352.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**



### 6.353

#### activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller Class Reference 1921

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3393).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ExceptionResponseMarshaller.h**

### 6.353 activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1919).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller**:

#### Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.353.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1919).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.353.2 Constructor & Destructor Documentation

6.353.2.1 **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::ExceptionResponseMarshaller**  
( ) [inline]

6.353.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::~~ExceptionResponseMarshaller**  
( ) [inline, virtual]

### 6.353.3 Member Function Documentation

6.353.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3410).

6.353

activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller

Class Reference

1923

6.353.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3410).

6.353.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::looseMarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3410).

6.353.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::looseUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3411).

**6.353.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3412).

**6.353.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3412).

**6.353.3.7** virtual void **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3413).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h`

**6.354 decaf::util::concurrent::ExecutionException Class Reference**

```
#include <src/main/decaf/util/concurrent/ExecutionException.h>
```

Inheritance diagram for **decaf::util::concurrent::ExecutionException**:

**Public Member Functions**

- **ExecutionException** () throw ()  
*Default Constructor.*
- **ExecutionException** (const **decaf::lang::Exception** &ex) throw ()  
*Conversion Constructor from some other Exception.*

- **ExecutionException** (const **ExecutionException** &ex) throw ()  
*Copy Constructor.*
- **ExecutionException** (const std::exception \***cause**) throw ()  
*Constructor.*
- **ExecutionException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **ExecutionException** (const char \*file, const int lineNumber, const std::exception \***cause**, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **ExecutionException** \* **clone** () const  
*Clones this exception.*
- virtual ~**ExecutionException** () throw ()

### 6.354.1 Constructor & Destructor Documentation

6.354.1.1 **decaf::util::concurrent::ExecutionException::ExecutionException ( ) throw ()**  
[inline]

Default Constructor.

6.354.1.2 **decaf::util::concurrent::ExecutionException::ExecutionException ( const decaf::lang::Exception & ex ) throw ()** [inline]

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	- An exception that should become this type of Exception
-----------	--

6.354.1.3 **decaf::util::concurrent::ExecutionException::ExecutionException ( const ExecutionException & ex ) throw ()** [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	- The Exception to copy in this new instance.
-----------	---

**6.354.1.4** decaf::util::concurrent::ExecutionException::ExecutionException ( const std::exception \* *cause* ) throw () [inline]

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.354.1.5** decaf::util::concurrent::ExecutionException::ExecutionException ( const char \* *file*, const int *lineNumber*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.
<i>msg</i>	- The message to report
...	- The list of primitives that are formatted into the message

**6.354.1.6** decaf::util::concurrent::ExecutionException::ExecutionException ( const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.
<i>cause</i>	- The exception that was the cause for this one to be thrown.
<i>msg</i>	- The message to report
...	- list of primitives that are formatted into the message

6.354.1.7 `virtual decaf::util::concurrent::ExecutionException::~~ExecutionException ( ) throw ( )`  
`[inline, virtual]`

## 6.354.2 Member Function Documentation

6.354.2.1 `virtual ExecutionException* decaf::util::concurrent::ExecutionException::clone ( ) const` `[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

a new instance of an exception that is a clone of this one.

Reimplemented from `decaf::lang::Exception` (p. 1889).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ExecutionException.h`

## 6.355 decaf::util::concurrent::Executor Class Reference

An object that executes submitted `decaf.lang Runnable` (p. 3418) tasks.

```
#include <src/main/decaf/util/concurrent/Executor.h>
```

Inheritance diagram for `decaf::util::concurrent::Executor`:

### Public Member Functions

- `virtual ~Executor ( )`
- `virtual void execute (Runnable *command)=0 throw ( decaf::util::concurrent::RejectedExecutionException decaf::lang::exceptions::NullPointerException )`

*Executes the given command at some time in the future.*

### 6.355.1 Detailed Description

An object that executes submitted `decaf.lang Runnable` (p. 3418) tasks. This interface provides a way of decoupling task submission from the mechanics of how each task will be run, including details of thread use, scheduling, etc. An `Executor` (p. 1926) is normally used instead of explicitly creating threads. For example, rather than invoking `new Thread (new RunnableTask ( ) ) .start ( )` for each of a set of tasks, you might use:



```
Executor (p.1926) executor = anExecutor;
executor->execute( new RunnableTask1() );
executor->execute( new RunnableTask2() );
...
```

However, the **Executor** (p.1926) interface does not strictly require that execution be asynchronous. In the simplest case, an executor can run the submitted task immediately in the caller's thread:

```
class DirectExecutor : public Executor (p.1926) {
public:

    void execute( Runnable* r ) (p.1928) {
        r->run();
    }

}
```

More typically, tasks are executed in some thread other than the caller's thread. The executor below spawns a new thread for each task.

```
class ThreadPerTaskExecutor : public Executor (p.1926) {
public:
    std::vector<Thread*> threads;

    void execute( Runnable* r ) (p.1928) {
        threads.push_back( new Thread( r ) );
        threads.rbegin()->start();
    }

}
```

The **Executor** (p.1926) implementations provided in this package implement **decaf.util.concurrent.ExecutorService** (p.1928), which is a more extensive interface. The **decaf.util.concurrent.ThreadPoolExecutor** (p.??) class provides an extensible thread pool implementation. The **decaf.util.concurrentExecutor** (p.??) class provides convenient factory methods for these Executors.

Since

1.0

## 6.355.2 Constructor & Destructor Documentation

6.355.2.1 `virtual decaf::util::concurrent::Executor::~~Executor ( ) [inline, virtual]`

## 6.355.3 Member Function Documentation

6.355.3.1 `virtual void decaf::util::concurrent::Executor::execute ( Runnable * command ) throw ( decaf::util::concurrent::RejectedExecutionException, decaf::lang::exceptions::NullPointerException ) [pure virtual]`

Executes the given command at some time in the future.

The command may execute in a new thread, in a pooled thread, or in the calling thread, at the discretion of the **Executor** (p. 1926) implementation.

### Parameters

<i>command</i>	the runnable task
----------------	-------------------

### Exceptions

<i>RejectedExecutionException</i> (p. 3280)	if this task cannot be accepted for execution.
<i>NullPointerException</i>	if command is null

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Executor.h`

## 6.356 decaf::util::concurrent::ExecutorService Class Reference

An **Executor** (p. 1926) that provides methods to manage termination and methods that can produce a **Future** (p. 2029) for tracking progress of one or more asynchronous tasks.

```
#include <src/main/decaf/util/concurrent/ExecutorService.h>
```

Inheritance diagram for `decaf::util::concurrent::ExecutorService`:

### Public Member Functions

- `virtual ~ExecutorService ()`
- `bool awaitTermination (long long timeout, const TimeUnit &unit)=0 throw ( decaf::lang::exceptions::InterruptedException )`

*Blocks until all tasks have completed execution after a shutdown request, or the timeout occurs, or the current thread is interrupted, whichever happens first.*

### 6.356.1 Detailed Description

An **Executor** (p. 1926) that provides methods to manage termination and methods that can produce a **Future** (p. 2029) for tracking progress of one or more asynchronous tasks. An **ExecutorService** (p. 1928) can be shut down, which will cause it to reject new tasks. Two different methods are provided for shutting down an **ExecutorService** (p. 1928). The shutdown() method will allow previously submitted tasks to execute before terminating, while the shutdownNow() method prevents waiting tasks from starting and attempts to stop currently executing tasks. Upon termination, an executor has no tasks actively executing, no tasks awaiting execution, and no new tasks can be submitted. An unused **ExecutorService** (p. 1928) should be shut down to allow reclamation of its resources.

Method submit extends base method **Executor.execute** (p. 1928)(**decaf.lang Runnable** (p. 3418)) by creating and returning a **Future** (p. 2029) that can be used to cancel execution and/or wait for completion. Methods invokeAny and invokeAll perform the most commonly useful forms of bulk execution, executing a collection of tasks and then waiting for at least one, or all, to complete. (Class ExecutorCompletionService can be used to write customized variants of these methods.)

The Executors class provides factory methods for the executor services provided in this package.

Since

1.0

### 6.356.2 Constructor & Destructor Documentation

6.356.2.1 virtual decaf::util::concurrent::ExecutorService::~~ExecutorService ( )  
[inline, virtual]

### 6.356.3 Member Function Documentation

6.356.3.1 bool decaf::util::concurrent::ExecutorService::awaitTermination  
( long long *timeout*, const TimeUnit & *unit* ) throw ( **decaf::lang::exceptions::InterruptedException** ) [pure virtual]

Blocks until all tasks have completed execution after a shutdown request, or the timeout occurs, or the current thread is interrupted, whichever happens first.

#### Parameters

<i>timeout</i>	The amount of time to wait before timing out the Wait operation.
<i>unit</i>	The Units that comprise the timeout value.

**Returns**

true if the executor terminated before the given timeout value elapsed.

**Exceptions**

<i>InterruptedException</i>	- if interrupted while waiting.
-----------------------------	---------------------------------

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ExecutorService.h**

## 6.357 activemq::transport::failover::FailoverTransport Class Reference

```
#include <src/main/activemq/transport/failover/FailoverTransport.h>
```

Inheritance diagram for activemq::transport::failover::FailoverTransport:

**Public Member Functions**

- **FailoverTransport** ()
- virtual **~FailoverTransport** ()
- void **reconnect** ()  
*Indicates that the **Transport** (p. 3996) needs to reconnect to another URI in its list.*
- void **add** (const std::string &uri)  
*Adds a New URI to the List of URIs this transport can Connect to.*
- virtual void **addURI** (const **List**< **URI** > &uris)  
*Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3996) is a composite of.*
- virtual void **removeURI** (const **List**< **URI** > &uris)  
*Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3996) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3996) should result in that **Transport** (p. 3996) being disposed of.*
- virtual void **start** () throw ( decaf::io::IOException )  
*Starts this transport object and creates the thread for polling on the input stream for commands.*
- virtual void **stop** () throw ( decaf::io::IOException )  
*Stop the **Transport** (p. 3996).*

- virtual void **close** () throw ( decaf::io::IOException )

*Stops the polling thread and closes the streams.*

- virtual void **oneway** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Sends a one-way command.*

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Sends the given command to the broker and then waits for the response.*

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Sends the given command to the broker and then waits for the response.*

- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat AMQCPP\_UNUSED)

*Sets the WireFormat instance to use.*

- virtual void **setTransportListener** (**TransportListener** \*listener)

*Sets the observer of asynchronous events from this transport.*

- virtual **TransportListener** \* **getTransportListener** () const

*Gets the observer of asynchronous exceptions from this transport.*

- virtual bool **isFaultTolerant** () const

*Is this **Transport** (p. 3996) fault tolerant, meaning that it will reconnect to a broker on disconnect.*

- virtual bool **isConnected** () const

*Is the **Transport** (p. 3996) Connected to its Broker.*

- virtual bool **isClosed** () const

*Has the **Transport** (p. 3996) been shutdown and no longer usable.*

- bool **isInitialized** () const

*Returns true if the **Transport** (p. 3996) has been initialized by a BrokerInfo command.*

- void **setInitialized** (bool value)

*Sets the initialized state of this **Transport** (p. 3996) to true.*

- virtual **Transport** \* **narrow** (const std::type\_info &typeId)

*Narrows down a Chain of Transports to a specific **Transport** (p.3996) to allow a higher level transport to skip intermediate Transports in certain circumstances.*

- virtual std::string **getRemoteAddress** () const
- virtual bool **isPending** () const
- virtual bool **iterate** ()

*Performs the actual Reconnect operation for the **FailoverTransport** (p. 1930), when a connection is made this method returns false to indicate it doesn't need to run again, otherwise it returns true to indicate its still trying to connect.*

- virtual void **reconnect** (const **decaf::net::URI** &uri) throw ( decaf::io::IOException )

*reconnect to another location*

- long long **getTimeout** () const
- void **setTimeout** (long long value)
- long long **getInitialReconnectDelay** () const
- void **setInitialReconnectDelay** (long long value)
- long long **getMaxReconnectDelay** () const
- void **setMaxReconnectDelay** (long long value)
- long long **getBackOffMultiplier** () const
- void **setBackOffMultiplier** (long long value)
- bool **isUseExponentialBackOff** () const
- void **setUseExponentialBackOff** (bool value)
- bool **isRandomize** () const
- void **setRandomize** (bool value)
- int **getMaxReconnectAttempts** () const
- void **setMaxReconnectAttempts** (int value)
- int **getStartupMaxReconnectAttempts** () const
- void **setStartupMaxReconnectAttempts** (int value)
- long long **getReconnectDelay** () const
- void **setReconnectDelay** (long long value)
- bool **isBackup** () const
- void **setBackup** (bool value)
- int **getBackupPoolSize** () const
- void **setBackupPoolSize** (int value)
- bool **isTrackMessages** () const
- void **setTrackMessages** (bool value)
- bool **isTrackTransactionProducers** () const
- void **setTrackTransactionProducers** (bool value)
- int **getMaxCacheSize** () const
- void **setMaxCacheSize** (int value)
- void **setConnectionInterruptProcessingComplete** (const **Pointer**< **commands::ConnectionId** > &connectionId)

## Protected Member Functions

- void **restoreTransport** (const **Pointer**< **Transport** > &transport) throw ( decaf::io::IOException )  
*Given a **Transport** (p. 3996) restore the state of the Client's connection to the Broker using the data accumulated in the State Tracker.*
- void **handleTransportFailure** (const decaf::lang::Exception &error) throw ( decaf::lang::Exception )  
*Called when this class' **TransportListener** (p. 4013) is notified of a Failure.*

## Friends

- class **FailoverTransportListener**

## 6.357.1 Constructor & Destructor Documentation

6.357.1.1 **activemq::transport::failover::FailoverTransport::FailoverTransport** ( )

6.357.1.2 **virtual activemq::transport::failover::FailoverTransport::~~FailoverTransport** ( )  
 [virtual]

## 6.357.2 Member Function Documentation

6.357.2.1 **void activemq::transport::failover::FailoverTransport::add** ( const std::string & *uri* )

Adds a New URI to the List of URIs this transport can Connect to.

### Parameters

<i>uri</i>	A String version of a URI to add to the URIs to failover to.
------------	--

6.357.2.2 **virtual void activemq::transport::failover::FailoverTransport::addURI** ( const List< **URI** > & *uris* ) [virtual]

Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3996) is a composite of.

### Parameters

<i>uris</i>	The new URIs to add to the set this composite maintains.
-------------	--

Implements **activemq::transport::CompositeTransport** (p. 1259).

**6.357.2.3** `virtual void activemq::transport::failover::FailoverTransport::close ( ) throw ( decaf::io::IOException )` `[virtual]`

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

#### Exceptions

<i>IOException</i>	if errors occur.
--------------------	------------------

Implements **decaf::io::Closeable** (p. 1181).

**6.357.2.4** `long long activemq::transport::failover::FailoverTransport::getBackOffMultiplier ( ) const` `[inline]`

**6.357.2.5** `int activemq::transport::failover::FailoverTransport::getBackupPoolSize ( ) const` `[inline]`

**6.357.2.6** `long long activemq::transport::failover::FailoverTransport::getInitialReconnectDelay ( ) const` `[inline]`

**6.357.2.7** `int activemq::transport::failover::FailoverTransport::getMaxCacheSize ( ) const` `[inline]`

**6.357.2.8** `int activemq::transport::failover::FailoverTransport::getMaxReconnectAttempts ( ) const` `[inline]`

**6.357.2.9** `long long activemq::transport::failover::FailoverTransport::getMaxReconnectDelay ( ) const` `[inline]`

**6.357.2.10** `long long activemq::transport::failover::FailoverTransport::getReconnectDelay ( ) const` `[inline]`

**6.357.2.11** `virtual std::string activemq::transport::failover::FailoverTransport::getRemoteAddress ( ) const` `[virtual]`

#### Returns

the remote address for this connection

Implements **activemq::transport::Transport** (p. 3998).



6.357.2.12 `int activemq::transport::failover::FailoverTransport::getStartupMaxReconnectAttempts ( ) const [inline]`

6.357.2.13 `long long activemq::transport::failover::FailoverTransport::getTimeout ( ) const [inline]`

6.357.2.14 `virtual TransportListener* activemq::transport::failover::FailoverTransport::getTransportListener ( ) const [virtual]`

Gets the observer of asynchronous exceptions from this transport.

#### Returns

The listener of transport events.

Implements **activemq::transport::Transport** (p. 3998).

6.357.2.15 `void activemq::transport::failover::FailoverTransport::handleTransportFailure ( const decaf::lang::Exception & error ) throw ( decaf::lang::Exception ) [protected]`

Called when this class' **TransportListener** (p. 4013) is notified of a Failure.

#### Parameters

<i>error</i>	- The CMS Exception that was thrown.
--------------	--------------------------------------

#### Exceptions

<i>Exception</i>	if an error occurs.
------------------	---------------------

6.357.2.16 `bool activemq::transport::failover::FailoverTransport::isBackup ( ) const [inline]`

6.357.2.17 `virtual bool activemq::transport::failover::FailoverTransport::isClosed ( ) const [inline, virtual]`

Has the **Transport** (p. 3996) been shutdown and no longer usable.

#### Returns

true if the **Transport** (p. 3996)

Implements **activemq::transport::Transport** (p. 3998).

**6.357.2.18** `virtual bool activemq::transport::failover::FailoverTransport::isConnected ( ) const`  
`[inline, virtual]`

Is the **Transport** (p. 3996) Connected to its Broker.

#### Returns

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 3998).

**6.357.2.19** `virtual bool activemq::transport::failover::FailoverTransport::isFaultTolerant ( )`  
`const [inline, virtual]`

Is this **Transport** (p. 3996) fault tolerant, meaning that it will reconnect to a broker on disconnect.

#### Returns

true if the **Transport** (p. 3996) is fault tolerant.

Implements **activemq::transport::Transport** (p. 3999).

**6.357.2.20** `bool activemq::transport::failover::FailoverTransport::isInitialized ( ) const`  
`[inline]`

Returns true if the **Transport** (p. 3996) has been initialized by a BrokerInfo command.

#### Returns

true if the **Transport** (p. 3996) has been initialized by a BrokerInfo command.

**6.357.2.21** `virtual bool activemq::transport::failover::FailoverTransport::isPending ( ) const`  
`[virtual]`

#### Returns

true if there is a need for the iterate method to be called by this classes task runner.

Implements **activemq::threads::CompositeTask** (p. 1255).

- 6.357.2.22 `bool activemq::transport::failover::FailoverTransport::isRandomize ( ) const`  
[inline]
- 6.357.2.23 `bool activemq::transport::failover::FailoverTransport::isTrackMessages ( ) const`  
[inline]
- 6.357.2.24 `bool activemq::transport::failover::FailoverTransport::isTrackTransactionProducers ( ) const` [inline]
- 6.357.2.25 `bool activemq::transport::failover::FailoverTransport::isUseExponentialBackOff ( ) const` [inline]
- 6.357.2.26 `virtual bool activemq::transport::failover::FailoverTransport::iterate ( )`  
[virtual]

Performs the actual Reconnect operation for the **FailoverTransport** (p. 1930), when a connection is made this method returns false to indicate it doesn't need to run again, otherwise it returns true to indicate its still trying to connect.

#### Returns

false to indicate a connection, true to indicate it needs to try again.

Implements **activemq::threads::Task** (p. 3847).

- 6.357.2.27 `virtual Transport* activemq::transport::failover::FailoverTransport::narrow ( const std::type_info & typeId )` [inline, virtual]

Narrows down a Chain of Transports to a specific **Transport** (p. 3996) to allow a higher level transport to skip intermediate Transports in certain circumstances.

#### Parameters

<i>typeId</i>	- The type_info of the Object we are searching for.
---------------	---

#### Returns

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 3999).

References **activemq::transport::Transport::narrow()**.

- 6.357.2.28 `virtual void activemq::transport::failover::FailoverTransport::oneway ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )`  
[virtual]

Sends a one-way command.

Does not wait for any response from the broker.

#### Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

#### Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 3999).

**6.357.2.29** `void activemq::transport::failover::FailoverTransport::reconnect ( )`

Indicates that the **Transport** (p. 3996) needs to reconnect to another URI in its list.

**6.357.2.30** `virtual void activemq::transport::failover::FailoverTransport::reconnect ( const decaf::net::URI & uri ) throw ( decaf::io::IOException ) [virtual]`

reconnect to another location

#### Parameters

<i>uri</i>	
------------	--

#### Exceptions

<i>IOException</i>	on failure of if not supported
--------------------	--------------------------------

Implements **activemq::transport::Transport** (p. 4000).

**6.357.2.31** `virtual void activemq::transport::failover::FailoverTransport::removeURI ( const List< URI > & uris ) [virtual]`

Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3996) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3996) should result in that **Transport** (p. 3996) being disposed of.

#### Parameters

<i>uris</i>	The new URIs to remove to the set this composite maintains.
-------------	---

Implements **activemq::transport::CompositeTransport** (p. 1260).

**6.357.2.32** virtual **Pointer<Response>** **activemq::transport::failover::FailoverTransport::request** ( const **Pointer<Command>** & *command* ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]

Sends the given command to the broker and then waits for the response.

#### Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

#### Returns

the response from the broker.

#### Exceptions

<i>IOException</i>	if an exception occurs during the read of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 4000).

**6.357.2.33** virtual **Pointer<Response>** **activemq::transport::failover::FailoverTransport::request** ( const **Pointer<Command>** & *command*, unsigned int *timeout* ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]

Sends the given command to the broker and then waits for the response.

#### Parameters

<i>command</i>	- The command to be sent.
<i>timeout</i>	- The time to wait for this response.

#### Returns

the response from the broker.

#### Exceptions

<i>IOException</i>	if an exception occurs during the read of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 4001).

6.357.2.34 `void activemq::transport::failover::FailoverTransport::restoreTransport ( const  
Pointer< Transport > & transport ) throw ( decaf::io::IOException )  
[protected]`

Given a **Transport** (p. 3996) restore the state of the Client's connection to the Broker using the data accumulated in the State Tracker.

#### Parameters

<i>transport</i>	The new <b>Transport</b> (p. 3996) connected to the Broker.
------------------	---

#### Exceptions

<i>IOException</i>	if an errors occurs while restoring the old state.
--------------------	--

6.357.2.35 `void activemq::transport::failover::FailoverTransport::setBackOffMultiplier ( long  
long value ) [inline]`

6.357.2.36 `void activemq::transport::failover::FailoverTransport::setBackup ( bool value )  
[inline]`

6.357.2.37 `void activemq::transport::failover::FailoverTransport::setBackupPoolSize ( int value  
) [inline]`

6.357.2.38 `void activemq::transport::failover::FailoverTransport::setConnectionInterruptProcessingComplete  
( const Pointer< commands::ConnectionId > & connectionId )`

6.357.2.39 `void activemq::transport::failover::FailoverTransport::setInitialized ( bool value )  
[inline]`

Sets the initialized state of this **Transport** (p. 3996) to true.

#### Parameters

<i>value</i>	- true if this <b>Transport</b> (p. 3996) has been initialized.
--------------	---

- 6.357.2.40 `void activemq::transport::failover::FailoverTransport::setInitialReconnectDelay ( long long value ) [inline]`
- 6.357.2.41 `void activemq::transport::failover::FailoverTransport::setMaxCacheSize ( int value ) [inline]`
- 6.357.2.42 `void activemq::transport::failover::FailoverTransport::setMaxReconnectAttempts ( int value ) [inline]`
- 6.357.2.43 `void activemq::transport::failover::FailoverTransport::setMaxReconnectDelay ( long long value ) [inline]`
- 6.357.2.44 `void activemq::transport::failover::FailoverTransport::setRandomize ( bool value ) [inline]`
- 6.357.2.45 `void activemq::transport::failover::FailoverTransport::setReconnectDelay ( long long value ) [inline]`
- 6.357.2.46 `void activemq::transport::failover::FailoverTransport::setStartupMaxReconnectAttempts ( int value ) [inline]`
- 6.357.2.47 `void activemq::transport::failover::FailoverTransport::setTimeout ( long long value ) [inline]`
- 6.357.2.48 `void activemq::transport::failover::FailoverTransport::setTrackMessages ( bool value ) [inline]`
- 6.357.2.49 `void activemq::transport::failover::FailoverTransport::setTrackTransactionProducers ( bool value ) [inline]`
- 6.357.2.50 `virtual void activemq::transport::failover::FailoverTransport::setTransportListener ( TransportListener * listener ) [virtual]`

Sets the observer of asynchronous events from this transport.

#### Parameters

<i>listener</i>	the listener of transport events.
-----------------	-----------------------------------

Implements `activemq::transport::Transport` (p. 4001).

- 6.357.2.51 `void activemq::transport::failover::FailoverTransport::setUseExponentialBackOff ( bool value ) [inline]`
- 6.357.2.52 `virtual void activemq::transport::failover::FailoverTransport::setWireFormat ( const Pointer< wireformat::WireFormat > &wireFormat AMQCPP_UNUSED ) [inline, virtual]`

Sets the WireFormat instance to use.

**Parameters**

<i>wireFormat</i>	The WireFormat the object used to encode / decode commands.
-------------------	---

**6.357.2.53** `virtual void activemq::transport::failover::FailoverTransport::start ( ) throw ( decaf::io::IOException )` [virtual]

Starts this transport object and creates the thread for polling on the input stream for commands.

If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

**Exceptions**

<i>IOException</i>	if an error occurs or if this transport has already been closed.
--------------------	--

Implements **activemq::transport::Transport** (p. 4002).

**6.357.2.54** `virtual void activemq::transport::failover::FailoverTransport::stop ( ) throw ( decaf::io::IOException )` [virtual]

Stop the **Transport** (p. 3996).

**Exceptions**

<i>IOException</i>	if an error occurs while stopping the <b>Transport</b> (p. 3996).
--------------------	---

Implements **activemq::transport::Transport** (p. 4002).

**6.357.3 Friends And Related Function Documentation**

**6.357.3.1** `friend class FailoverTransportListener` [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**FailoverTransport.h**

**6.358 activemq::transport::failover::FailoverTransportFactory Class Reference**

Creates an instance of a **FailoverTransport** (p. 1930).

```
#include <src/main/activemq/transport/failover/FailoverTransportFactory.h>
```



## 6.358 `activemq::transport::failover::FailoverTransportFactory` Class Reference 1045

Inheritance diagram for `activemq::transport::failover::FailoverTransportFactory`:

### Public Member Functions

- virtual `~FailoverTransportFactory()`
- virtual `Pointer< Transport > create` (const `decaf::net::URI` &location) throw ( exceptions::ActiveMQException )

*Creates a fully configured **Transport** (p. 3996) instance which could be a chain of filters and transports.*

- virtual `Pointer< Transport > createComposite` (const `decaf::net::URI` &location) throw ( exceptions::ActiveMQException )

*Creates a slimed down **Transport** (p. 3996) instance which can be used in composite transport instances.*

### Protected Member Functions

- virtual `Pointer< Transport > doCreateComposite` (const `decaf::net::URI` &location, const `decaf::util::Properties` &properties) throw ( exceptions::ActiveMQException )

*Creates a slimed down **Transport** (p. 3996) instance which can be used in composite transport instances.*

#### 6.358.1 Detailed Description

Creates an instance of a `FailoverTransport` (p. 1930).

Since

3.0

## 6.358.2 Constructor & Destructor Documentation

6.358.2.1 `virtual activemq::transport::failover::FailoverTransportFactory::~~FailoverTransportFactory ( ) [inline, virtual]`

## 6.358.3 Member Function Documentation

6.358.3.1 `virtual Pointer<Transport> activemq::transport::failover::FailoverTransportFactory::create ( const decaf::net::URI & location ) throw ( exceptions::ActiveMQException ) [virtual]`

Creates a fully configured **Transport** (p. 3996) instance which could be a chain of filters and transports.

### Parameters

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

### Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implements `activemq::transport::TransportFactory` (p. 4003).

6.358.3.2 `virtual Pointer<Transport> activemq::transport::failover::FailoverTransportFactory::createComposite ( const decaf::net::URI & location ) throw ( exceptions::ActiveMQException ) [virtual]`

Creates a slimmed down **Transport** (p. 3996) instance which can be used in composite transport instances.

### Parameters

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

### Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implements `activemq::transport::TransportFactory` (p. 4004).

## 6.359 activemq::transport::failover::FailoverTransportListener Class Reference

**6.358.3.3** virtual **Pointer**<**Transport**> **activemq::transport::failover::FailoverTransportFactory::doCreateComposite** ( const **decaf::net::URI** & *location*, const **decaf::util::Properties** & *properties* ) throw ( **exceptions::ActiveMQException** ) [protected, virtual]

Creates a slimed down **Transport** (p. 3996) instance which can be used in composite transport instances.

### Parameters

<i>location</i>	- URI location to connect to.
<i>properties</i>	- Properties to apply to the transport.

### Returns

Pointer to a new **FailoverTransport** (p. 1930) instance.

### Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**FailoverTransportFactory.h**

## 6.359 activemq::transport::failover::FailoverTransportListener Class Reference

Utility class used by the **Transport** (p. 3996) to perform the work of responding to events from the active **Transport** (p. 3996).

```
#include <src/main/activemq/transport/failover/FailoverTransportListener.h>
```

Inheritance diagram for **activemq::transport::failover::FailoverTransportListener**:

### Public Member Functions

- **FailoverTransportListener** (**FailoverTransport** \*parent)
- virtual ~**FailoverTransportListener** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)  
*Event handler for the receipt of a command.*
- virtual void **onException** (const **decaf::lang::Exception** &ex)  
*Event handler for an exception from a command transport.*
- virtual void **transportInterrupted** ()

*The transport has suffered an interruption from which it hopes to recover.*

- virtual void **transportResumed** ()

*The transport has resumed after an interruption.*

### 6.359.1 Detailed Description

Utility class used by the **Transport** (p. 3996) to perform the work of responding to events from the active **Transport** (p. 3996).

Since

3.0

### 6.359.2 Constructor & Destructor Documentation

6.359.2.1 **activemq::transport::failover::FailoverTransportListener::FailoverTransportListener ( FailoverTransport \* parent )**

6.359.2.2 **virtual activemq::transport::failover::FailoverTransportListener::~~FailoverTransportListener ( ) [virtual]**

### 6.359.3 Member Function Documentation

6.359.3.1 **virtual void activemq::transport::failover::FailoverTransportListener::onCommand ( const Pointer< Command > & command ) [virtual]**

Event handler for the receipt of a command.

The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3996) deletes the command upon receipt.

#### Parameters

<i>command</i>	the received command object.
----------------	------------------------------

Implements **activemq::transport::TransportListener** (p. 4014).

6.359.3.2 **virtual void activemq::transport::failover::FailoverTransportListener::onException ( const decaf::lang::Exception & ex ) [virtual]**

Event handler for an exception from a command transport.

#### Parameters

<i>ex</i>	The exception.
-----------	----------------

Implements **activemq::transport::TransportListener** (p. 4015).

**6.359.3.3** `virtual void activemq::transport::failover::FailoverTransportListener::transportInterrupted ( ) [virtual]`

The transport has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 4015).

**6.359.3.4** `virtual void activemq::transport::failover::FailoverTransportListener::transportResumed ( ) [virtual]`

The transport has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 4015).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/FailoverTransportListener.h`

## 6.360 decaf::io::FileDescriptor Class Reference

This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.

```
#include <src/main/decaf/io/FileDescriptor.h>
```

Inheritance diagram for decaf::io::FileDescriptor:

### Public Member Functions

- **FileDescriptor** ()
- virtual **~FileDescriptor** ()
- void **sync** ()  
*Force any/all buffered data for this **FileDescriptor** (p. 1947) to be flushed to the underlying device.*
- bool **valid** ()  
*Indicates whether the File Descriptor is valid.*

### Static Public Attributes

- static **FileDescriptor** **in**  
*A handle to the standard input stream.*

- static **FileDescriptor out**  
*A handle to the standard output stream.*
- static **FileDescriptor err**  
*A handle to the standard error stream.*

### Protected Member Functions

- **FileDescriptor** (long value, bool **readonly**)

### Protected Attributes

- long **descriptor**
- bool **readonly**

### 6.360.1 Detailed Description

This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.

#### Since

1.0

### 6.360.2 Constructor & Destructor Documentation

6.360.2.1 **decaf::io::FileDescriptor::FileDescriptor** ( long value, bool readonly )  
[protected]

6.360.2.2 **decaf::io::FileDescriptor::FileDescriptor** ( )

6.360.2.3 **virtual decaf::io::FileDescriptor::~~FileDescriptor** ( ) [virtual]

### 6.360.3 Member Function Documentation

6.360.3.1 **void decaf::io::FileDescriptor::sync** ( )

Force any/all buffered data for this **FileDescriptor** (p. 1947) to be flushed to the underlying device.

This method blocks until all data is flushed to the underlying device and is used to place the device into a known state. In the case of data that is buffered at a higher level such as a **BufferedOutputStream** (p. 949) the stream must first be flushed before this method can force the data to be sent to the output device.

### 6.360.3.2 bool decaf::io::FileDescriptor::valid ( )

Indicates whether the File Descriptor is valid.

#### Returns

true for a valid descriptor such as open socket or file, false otherwise.

## 6.360.4 Field Documentation

### 6.360.4.1 long decaf::io::FileDescriptor::descriptor [protected]

### 6.360.4.2 FileDescriptor decaf::io::FileDescriptor::err [static]

A handle to the standard error stream.

Usually, this file descriptor is not used directly, but rather via the output stream known as System::err.

### 6.360.4.3 FileDescriptor decaf::io::FileDescriptor::in [static]

A handle to the standard input stream.

Usually, this file descriptor is not used directly, but rather via the input stream known as System::in.

### 6.360.4.4 FileDescriptor decaf::io::FileDescriptor::out [static]

A handle to the standard output stream.

Usually, this file descriptor is not used directly, but rather via the output stream known as System::out.

### 6.360.4.5 bool decaf::io::FileDescriptor::readonly [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/io/FileDescriptor.h

## 6.361 decaf::util::logging::Filter Class Reference

A **Filter** (p. 1949) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.

```
#include <src/main/decaf/util/logging/Filter.h>
```

## Public Member Functions

- virtual `~Filter()`
- virtual `bool isLoggable (const LogRecord &record) const =0`

*Check if a given log record should be published.*

### 6.361.1 Detailed Description

A **Filter** (p. 1949) can be used to provide fine grain control over what is logged, beyond the control provided by log levels. Each **Logger** (p. 2461) and each **Handler** (p. 2042) can have a filter associated with it. The **Logger** (p. 2461) or **Handler** (p. 2042) will call the `isLoggable` method to check if a given **LogRecord** (p. 2487) should be published. If `isLoggable` returns false, the **LogRecord** (p. 2487) will be discarded.

### 6.361.2 Constructor & Destructor Documentation

6.361.2.1 `virtual decaf::util::logging::Filter::~Filter ( ) [inline, virtual]`

### 6.361.3 Member Function Documentation

6.361.3.1 `virtual bool decaf::util::logging::Filter::isLoggable ( const LogRecord & record ) const [pure virtual]`

Check if a given log record should be published.

#### Parameters

<i>record</i>	the <b>LogRecord</b> (p. 2487) to check.
---------------	--

#### Returns

true if the record is loggable.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/Filter.h`

## 6.362 decaf::io::FilterInputStream Class Reference

A **FilterInputStream** (p. 1950) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.

```
#include <src/main/decaf/io/FilterInputStream.h>
```



Inheritance diagram for decaf::io::FilterInputStream:

## Public Member Functions

- **FilterInputStream** (**InputStream** \***inputStream**, bool **own**=false)

Constructor to create a wrapped **InputStream** (p. 2105).

- virtual ~**FilterInputStream** ()
- virtual int **available** () const throw ( decaf::io::IOException )

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

### Returns

the number of bytes available on this input stream.

### Exceptions

<b>IOException</b> (p. 2210)	if an I/O error occurs.
------------------------------	-------------------------

- virtual void **close** () throw ( decaf::io::IOException )

Closes the **InputStream** (p. 2105) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2105) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

### Parameters

num	The number of bytes to skip.
-----	------------------------------

### Returns

total bytes skipped

### Exceptions

<b>IOException</b> (p. 2210)	if an I/O error occurs.
------------------------------	-------------------------

UnsupportedOperationException	if the concrete stream class does not support skipping bytes.
-------------------------------	---

- virtual void **mark** (int readLimit)

*Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.*

*If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.*

*Calling mark on a closed stream instance should have no effect.*

*The default implementation of this method does nothing.*

#### Parameters

readLimit	The max bytes read before marked position is invalid.
-----------	---

- virtual void **reset** () throw ( decaf::io::IOException )

*Repositions this stream to the position at the time the mark method was last called on this input stream.*

*If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2210) might be thrown. \* If such an **IOException** (p. 2210) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.*

*If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2210). \* If an **IOException** (p. 2210) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.*

*The default implementation of this method throws an **IOException** (p. 2210).*

#### Exceptions

<b>IOException</b> (p. 2210)	if an I/O error occurs.
------------------------------	-------------------------

- virtual bool **markSupported** () const

*Determines if this input stream supports the mark and reset methods.*

*Whether or not mark and reset are supported is an invariant property of a particular input stream instance.*

*The default implementation of this method returns false.*

#### Returns

*true if this stream instance supports marks*

## Protected Member Functions

- virtual int **doReadByte** () throw ( decaf::io::IOException )
- virtual int **doReadArray** (unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )
- virtual bool **isClosed** () const

### Protected Attributes

- **InputStream** \* **inputStream**
- bool **own**
- volatile bool **closed**

### 6.362.1 Detailed Description

A **FilterInputStream** (p. 1950) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality. The class **FilterInputStream** (p. 1950) itself simply overrides all methods of **InputStream** (p. 2105) with versions that pass all requests to the contained input stream. Subclasses of **FilterInputStream** (p. 1950) may further override some of these methods and may also provide additional methods and fields.

### 6.362.2 Constructor & Destructor Documentation

**6.362.2.1** decaf::io::FilterInputStream::FilterInputStream ( **InputStream** \* *inputStream*, bool *own* = false )

Constructor to create a wrapped **InputStream** (p. 2105).

#### Parameters

<i>inputStream</i>	The stream to wrap and filter.
<i>own</i>	Indicates if we own the stream object, defaults to false.

**6.362.2.2** virtual decaf::io::FilterInputStream::~~FilterInputStream ( ) [virtual]

### 6.362.3 Member Function Documentation

**6.362.3.1** virtual int decaf::io::FilterInputStream::available ( ) const throw ( decaf::io::IOException ) [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

**Returns**

the number of bytes available on this input stream.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
--	-------------------------

Reimplemented from **decaf::io::InputStream** (p. 2107).

Reimplemented in **decaf::io::BufferedInputStream** (p. 946), **decaf::io::PushbackInputStream** (p. 3234), and **decaf::util::zip::InflaterInputStream** (p. 2101).

**6.362.3.2** `virtual void decaf::io::FilterInputStream::close ( ) throw ( decaf::io::IOException ) [virtual]`

Closes the **InputStream** (p. 2105) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::InputStream** (p. 2108).

Reimplemented in **decaf::io::BufferedInputStream** (p. 946), and **decaf::util::zip::InflaterInputStream** (p. 2102).

**6.362.3.3** `virtual int decaf::io::FilterInputStream::doReadArray ( unsigned char * buffer, int size ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException ) [protected, virtual]`

Reimplemented from **decaf::io::InputStream** (p. 2108).

**6.362.3.4** `virtual int decaf::io::FilterInputStream::doReadArrayBounded ( unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException ) [protected, virtual]`

Reimplemented from **decaf::io::InputStream** (p. 2108).

Reimplemented in **activemq::io::LoggingInputStream** (p. 2475), **decaf::io::BufferedInputStream** (p. 946), **decaf::io::PushbackInputStream** (p. 3235), **decaf::util::zip::CheckedInputStream** (p. 1171), and **decaf::util::zip::InflaterInputStream** (p. 2102).

**6.362.3.5** `virtual int decaf::io::FilterInputStream::doReadByte ( ) throw ( decaf::io::IOException ) [protected, virtual]`

Implements **decaf::io::InputStream** (p. 2109).

Reimplemented in **activemq::io::LoggingInputStream** (p. 2475), **decaf::io::BufferedInputStream** (p. 946), **decaf::io::PushbackInputStream** (p. 3235), **decaf::util::zip::CheckedInputStream** (p. 1171), and **decaf::util::zip::InflaterInputStream** (p. 2102).

**6.362.3.6** `virtual bool decaf::io::FilterInputStream::isClosed ( ) const` [protected, virtual]

#### Returns

true if this stream has been closed.

**6.362.3.7** `virtual void decaf::io::FilterInputStream::mark ( int readLimit )` [virtual]

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

#### Parameters

<i>readLimit</i>	The max bytes read before marked position is invalid.
------------------	---

Reimplemented from **decaf::io::InputStream** (p. 2109).

Reimplemented in **decaf::io::BufferedInputStream** (p. 947), **decaf::io::PushbackInputStream** (p. 3235), and **decaf::util::zip::InflaterInputStream** (p. 2102).

**6.362.3.8** `virtual bool decaf::io::FilterInputStream::markSupported ( ) const` [virtual]

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

#### Returns

true if this stream instance supports marks

Reimplemented from **decaf::io::InputStream** (p. 2109).

Reimplemented in **decaf::io::BufferedInputStream** (p. 947), **decaf::io::PushbackInputStream** (p. 3235), and **decaf::util::zip::InflaterInputStream** (p. 2103).

**6.362.3.9** `virtual void decaf::io::FilterInputStream::reset ( ) throw ( decaf::io::IOException ) [virtual]`

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2210) might be thrown. \* If such an **IOException** (p. 2210) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2210). \* If an **IOException** (p. 2210) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2210).

#### Exceptions

<b>IOException</b> (p. 2210)	if an I/O error occurs.
---------------------------------	-------------------------

Reimplemented from **decaf::io::InputStream** (p. 2113).

Reimplemented in **decaf::io::BufferedInputStream** (p. 947), **decaf::io::PushbackInputStream** (p. 3236), and **decaf::util::zip::InflaterInputStream** (p. 2103).

**6.362.3.10** `virtual long long decaf::io::FilterInputStream::skip ( long long num ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2105) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

**Returns**

total bytes skipped

**Exceptions**

<i><b>IOException</b></i> (p. 2210)	if an I/O error occurs.
<i><b>UnsupportedOperationException</b></i>	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 2113).

Reimplemented in **decaf::io::BufferedInputStream** (p. 948), **decaf::io::PushbackInputStream** (p. 3236), **decaf::util::zip::CheckedInputStream** (p. 1171), and **decaf::util::zip::InflaterInputStream** (p. 2104).

**6.362.4 Field Documentation**

**6.362.4.1** `volatile bool decaf::io::FilterInputStream::closed` [protected]

**6.362.4.2** `InputStream* decaf::io::FilterInputStream::inputStream`  
[protected]

**6.362.4.3** `bool decaf::io::FilterInputStream::own` [protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/io/FilterInputStream.h`

**6.363 decaf::io::FilterOutputStream Class Reference**

This class is the superclass of all classes that filter output streams.

```
#include <src/main/decaf/io/FilterOutputStream.h>
```

Inheritance diagram for decaf::io::FilterOutputStream:

**Public Member Functions**

- **FilterOutputStream** (**OutputStream** \*outputStream, bool own=false)  
*Constructor, creates a wrapped output stream.*
- virtual **~FilterOutputStream** ()
- virtual void **flush** () throw ( decaf::io::IOException )  
*Flushes this stream by writing any buffered output to the underlying stream.*

*Exceptions*

<b>IOException</b> (p. 2210)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

*The default implementation of this method does nothing.*

- virtual void **close** () throw ( decaf::io::IOException )

*Closes this object and deallocates the appropriate resources.*

*The object is generally no longer usable after calling close.*

*Exceptions*

<b>IOException</b> (p. 2210)	<i>if an error occurs while closing.</i>
------------------------------	--

*The default implementation of this method does nothing.*

- virtual std::string **toString** () const

*Output a String representation of this object.*

*The default version of this method just prints the Class Name.*

*Returns*

*a string representation of the object.*

**Protected Member Functions**

- virtual void **doWriteByte** (unsigned char value) throw ( decaf::io::IOException )
- virtual void **doWriteArray** (const unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual bool **isClosed** () const

**Protected Attributes**

- **OutputStream \* outputStream**
- bool **own**
- volatile bool **closed**

**6.363.1 Detailed Description**

This class is the superclass of all classes that filter output streams. These streams sit on top of an already existing output stream (the underlying output stream) which it uses as its basic sink of data, but possibly transforming the data along the way or providing additional functionality.

The class **FilterOutputStream** (p. 1957) itself simply overrides all methods of **OutputStream** (p. 2991) with versions that pass all requests to the underlying output stream.



Subclasses of **FilterOutputStream** (p. 1957) may further override some of these methods as well as provide additional methods and fields.

Due to the lack of garbage collection in C++ a design decision was made to add a boolean parameter to the constructor indicating if the wrapped **InputStream** (p. 2105) is owned by this object. That way creation of the underlying stream can occur in a Java like way. Ex:

**DataOutputStream** (p. 1628) os = new **DataOutputStream** (p. 1628)( new **OutputStream**() (p. 2993), true )

### 6.363.2 Constructor & Destructor Documentation

6.363.2.1 **decaf::io::FilterOutputStream::FilterOutputStream ( OutputStream \* outputStream, bool own = false )**

Constructor, creates a wrapped output stream.

#### Parameters

<i>output-Stream</i>	the <b>OutputStream</b> (p. 2991) to wrap
<i>own</i>	If true, this object will control the lifetime of the output stream that it encapsulates.

6.363.2.2 **virtual decaf::io::FilterOutputStream::~FilterOutputStream ( ) [virtual]**

### 6.363.3 Member Function Documentation

6.363.3.1 **virtual void decaf::io::FilterOutputStream::close ( ) throw ( decaf::io::IOException ) [virtual]**

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

#### Exceptions

<b>IOException</b> (p. 2210)	if an error occurs while closing.
---------------------------------	-----------------------------------

The default implementation of this method does nothing.

The close method of **FilterOutputStream** (p. 1957) calls its flush method, and then calls the close method of its underlying output stream.

Reimplemented from **decaf::io::OutputStream** (p. 2993).

Reimplemented in **decaf::util::zip::DeflaterOutputStream** (p. 1772).

**6.363.3.2** `virtual void decaf::io::FilterOutputStream::doWriteArray ( const unsigned char * buffer, int size ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`  
`[protected, virtual]`

Reimplemented from **decaf::io::OutputStream** (p. 2994).

Reimplemented in **decaf::io::BufferedOutputStream** (p. 950).

**6.363.3.3** `virtual void decaf::io::FilterOutputStream::doWriteArrayBounded ( const unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`  
`[protected, virtual]`

Reimplemented from **decaf::io::OutputStream** (p. 2994).

Reimplemented in **activemq::io::LoggingOutputStream** (p. 2477), **decaf::io::BufferedOutputStream** (p. 950), **decaf::io::DataOutputStream** (p. 1630), **decaf::util::zip::CheckedOutputStream** (p. 1173), and **decaf::util::zip::DeflaterOutputStream** (p. 1773).

**6.363.3.4** `virtual void decaf::io::FilterOutputStream::doWriteByte ( unsigned char value ) throw ( decaf::io::IOException )` `[protected, virtual]`

Implements **decaf::io::OutputStream** (p. 2994).

Reimplemented in **activemq::io::LoggingOutputStream** (p. 2477), **decaf::io::BufferedOutputStream** (p. 950), **decaf::io::DataOutputStream** (p. 1630), **decaf::util::zip::CheckedOutputStream** (p. 1173), and **decaf::util::zip::DeflaterOutputStream** (p. 1773).

**6.363.3.5** `virtual void decaf::io::FilterOutputStream::flush ( ) throw ( decaf::io::IOException )` `[virtual]`

Flushes this stream by writing any buffered output to the underlying stream.

### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
--	-------------------------

The default implementation of this method does nothing.

The flush method of **FilterOutputStream** (p. 1957) calls the flush method of its underlying output stream.

Reimplemented from **decaf::io::OutputStream** (p. 2994).

Reimplemented in **decaf::io::BufferedOutputStream** (p. 951).

**6.363.3.6** `virtual bool decaf::io::FilterOutputStream::isClosed ( ) const` [protected, virtual]

#### Returns

true if this stream has been closed.

**6.363.3.7** `virtual std::string decaf::io::FilterOutputStream::toString ( ) const` [virtual]

Output a String representation of this object.

The default version of this method just prints the Class Name.

#### Returns

a string representation of the object.

The toString method of **FilterOutputStream** (p. 1957) calls the toString method of its underlying output stream.

Reimplemented from **decaf::io::OutputStream** (p. 2995).

### 6.363.4 Field Documentation

**6.363.4.1** `volatile bool decaf::io::FilterOutputStream::closed` [protected]

**6.363.4.2** `OutputStream* decaf::io::FilterOutputStream::outputStream` [protected]

**6.363.4.3** `bool decaf::io::FilterOutputStream::own` [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/io/**FilterOutputStream.h**

## 6.364 decaf::lang::Float Class Reference

```
#include <src/main/decaf/lang/Float.h>
```

Inheritance diagram for decaf::lang::Float:

### Public Member Functions

- **Float** (float value)
- **Float** (double value)

- **Float** (const std::string &value) throw ( exceptions::NumberFormatException )
- virtual ~**Float** ()
- virtual int **compareTo** (const **Float** &f) const  
*Compares this **Float** (p. 1961) instance with another.*
- bool **equals** (const **Float** &f) const
- virtual bool **operator==** (const **Float** &f) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Float** &f) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual int **compareTo** (const float &f) const  
*Compares this **Float** (p. 1961) instance with another.*
- bool **equals** (const float &f) const
- virtual bool **operator==** (const float &f) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const float &f) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- std::string **toString** () const
- virtual double **doubleValue** () const  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const  
*Answers the float value which the receiver represents.*
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual short **shortValue** () const  
*Answers the short value which the receiver represents.*
- virtual int **intValue** () const  
*Answers the int value which the receiver represents.*
- virtual long long **longValue** () const  
*Answers the long value which the receiver represents.*
- bool **isInfinite** () const
- bool **isNaN** () const

### Static Public Member Functions

- static int **compare** (float f1, float f2)  
*Compares the two specified double values.*
- static int **floatToIntBits** (float value)  
*Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout.*
- static int **floatToRawIntBits** (float value)  
*Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout, preserving Not-a-Number (NaN) values.*
- static float **intBitsToFloat** (int bits)  
*Returns the float value corresponding to a given bit representation.*
- static bool **isInfinite** (float value)
- static bool **isNaN** (float value)
- static float **parseFloat** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Returns a new float initialized to the value represented by the specified string, as performed by the valueOf method of class **Float** (p. 1961).*
- static std::string **toHexString** (float value)  
*Returns a hexadecimal string representation of the float argument.*
- static std::string **toString** (float value)  
*Returns a string representation of the float argument.*
- static **Float** **valueOf** (float value)  
*Returns a **Float** (p. 1961) instance representing the specified float value.*
- static **Float** **valueOf** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Returns a **Float** (p. 1961) instance that wraps a primitive float which is parsed from the string value passed.*

### Static Public Attributes

- static const int **SIZE** = 32  
*The size in bits of the primitive int type.*
- static const float **MAX\_VALUE**  
*The maximum value that the primitive type can hold.*
- static const float **MIN\_VALUE**

*The minimum value that the primitive type can hold.*

- static const float **NaN**  
*Constant for the Not a **Number** (p. 2918) Value.*
- static const float **POSITIVE\_INFINITY**  
*Constant for Positive Infinity.*
- static const float **NEGATIVE\_INFINITY**  
*Constant for Negative Infinity.*

### 6.364.1 Constructor & Destructor Documentation

#### 6.364.1.1 `decaf::lang::Float::Float ( float value )`

##### Parameters

<i>value</i>	- the primitive type to wrap
--------------	------------------------------

#### 6.364.1.2 `decaf::lang::Float::Float ( double value )`

##### Parameters

<i>value</i>	- the primitive type to wrap
--------------	------------------------------

#### 6.364.1.3 `decaf::lang::Float::Float ( const std::string & value ) throw ( exceptions::NumberFormatException )`

##### Parameters

<i>value</i>	- the string to convert to a primitive type to wrap
--------------	---

#### 6.364.1.4 `virtual decaf::lang::Float::~~Float ( ) [inline, virtual]`

### 6.364.2 Member Function Documentation

#### 6.364.2.1 `virtual unsigned char decaf::lang::Float::byteValue ( ) const [inline, virtual]`

Answers the byte value which the receiver represents.

##### Returns

byte the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2919).

#### 6.364.2.2 static int decaf::lang::Float::compare ( float *f1*, float *f2* ) [static]

Compares the two specified double values.

The sign of the integer value returned is the same as that of the integer that would be returned by the call: `Float( f1 ).compareTo( Float( f2 ) )`

##### Parameters

<i>f1</i>	- the first double to compare
<i>f2</i>	- the second double to compare

##### Returns

the value 0 if *d1* is numerically equal to *f2*; a value less than 0 if *f1* is numerically less than *f2*; and a value greater than 0 if *f1* is numerically greater than *f2*.

#### 6.364.2.3 virtual int decaf::lang::Float::compareTo ( const float & *f* ) const [virtual]

Compares this **Float** (p. 1961) instance with another.

##### Parameters

<i>f</i>	- the <b>Float</b> (p. 1961) instance to be compared
----------	--

##### Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< float > (p. 1249).

#### 6.364.2.4 virtual int decaf::lang::Float::compareTo ( const Float & *f* ) const [virtual]

Compares this **Float** (p. 1961) instance with another.

##### Parameters

<i>f</i>	- the <b>Float</b> (p. 1961) instance to be compared
----------	--

##### Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< Float > (p. 1249).

**6.364.2.5** `virtual double decaf::lang::Float::doubleValue ( ) const` `[inline, virtual]`

Answers the double value which the receiver represents.

#### Returns

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2919).

**6.364.2.6** `bool decaf::lang::Float::equals ( const Float & f ) const` `[inline, virtual]`

#### Parameters

<i>f</i>	- the <b>Float</b> (p. 1961) object to compare against.
----------	---

#### Returns

true if the two **Float** (p. 1961) Objects have the same value.

Implements **decaf::lang::Comparable**< **Float** > (p. 1250).

**6.364.2.7** `bool decaf::lang::Float::equals ( const float & f ) const` `[inline, virtual]`

#### Parameters

<i>f</i>	- the <b>Float</b> (p. 1961) object to compare against.
----------	---

#### Returns

true if the two **Float** (p. 1961) Objects have the same value.

Implements **decaf::lang::Comparable**< **float** > (p. 1250).

**6.364.2.8** `static int decaf::lang::Float::floatToIntBits ( float value )` `[static]`

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout.

Bit 31 (the bit that is selected by the mask 0x80000000) represents the sign of the floating-point number. Bits 30-23 (the bits that are selected by the mask 0x7f800000) represent the exponent. Bits 22-0 (the bits that are selected by the mask 0x007fffff) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7f800000. If the argument is negative infinity, the result is 0xff800000. If the argument is NaN, the result is 0x7fc00000.

In all cases, the result is an integer that, when given to the **intBitsToFloat(int)** (p. 1967) method, will produce a floating-point value the same as the argument to **floatToIntBits** (except all NaN values are collapsed to a single "canonical" NaN value).



**Parameters**

<i>value</i>	- the float to convert to int bits
--------------	------------------------------------

**Returns**

the int that holds the float's value

**6.364.2.9 static int decaf::lang::Float::floatToRawIntBits ( float *value* ) [static]**

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout, preserving Not-a-Number (NaN) values.

Bit 31 (the bit that is selected by the mask 0x80000000) represents the sign of the floating-point number. Bits 30-23 (the bits that are selected by the mask 0x7f800000) represent the exponent. Bits 22-0 (the bits that are selected by the mask 0x007fffff) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7f800000. If the argument is negative infinity, the result is 0xff800000. If the argument is NaN, the result is the integer representing the actual NaN value. Unlike the floatToIntBits method, intToRawIntBits does not collapse all the bit patterns encoding a NaN to a single "canonical" NaN value.

In all cases, the result is an integer that, when given to the **intBitsToFloat(int)** (p. 1967) method, will produce a floating-point value the same as the argument to floatToRawIntBits.

**Parameters**

<i>value</i>	The float to convert to a raw int.
--------------	------------------------------------

**Returns**

the raw int value of the float

**6.364.2.10 virtual float decaf::lang::Float::floatValue ( ) const [inline, virtual]**

Answers the float value which the receiver represents.

**Returns**

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2920).

**6.364.2.11 static float decaf::lang::Float::intBitsToFloat ( int *bits* ) [static]**

Returns the float value corresponding to a given bit representation.

The argument is considered to be a representation of a floating-point value according to the IEEE 754 floating-point "single format" bit layout.

If the argument is 0x7f800000, the result is positive infinity. If the argument is 0xff800000, the result is negative infinity. If the argument is any value in the range 0x7f800001 through 0x7fffffff or in the range 0xff800001 through 0xffffffff, the result is a NaN. No IEEE 754 floating-point operation provided by C++ can distinguish between two NaN values of the same type with different bit patterns. Distinct values of NaN are only distinguishable by use of the **Float::floatToRawIntBits** (p. 1967) method.

#### Parameters

<i>bits</i>	- the bits of the float encoded as a float
-------------	--

#### Returns

a new float created from the int bits.

**6.364.2.12** `virtual int decaf::lang::Float::intValue ( ) const` `[inline, virtual]`

Answers the int value which the receiver represents.

#### Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2920).

**6.364.2.13** `bool decaf::lang::Float::isInfinite ( ) const`

#### Returns

true if the float is equal to positive infinity.

**6.364.2.14** `static bool decaf::lang::Float::isInfinite ( float value )` `[static]`

#### Parameters

<i>value</i>	- The float to check.
--------------	-----------------------

#### Returns

true if the float is equal to infinity.

**6.364.2.15** `bool decaf::lang::Float::isNaN ( ) const`

#### Returns

true if the float is equal to NaN.

**6.364.2.16** `static bool decaf::lang::Float::isNaN ( float value ) [static]`

#### Parameters

<i>value</i>	- The float to check.
--------------	-----------------------

#### Returns

true if the float is equal to NaN.

**6.364.2.17** `virtual long long decaf::lang::Float::longValue ( ) const [inline, virtual]`

Answers the long value which the receiver represents.

#### Returns

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2920).

**6.364.2.18** `virtual bool decaf::lang::Float::operator< ( const float & f ) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>f</i>	- the value to be compared to this one.
----------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< float >** (p. 1250).

**6.364.2.19** `virtual bool decaf::lang::Float::operator< ( const Float & f ) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>f</i>	- the value to be compared to this one.
----------	---

**Returns**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Float** > (p. 1250).

**6.364.2.20** `virtual bool decaf::lang::Float::operator==( const Float & f ) const` [inline, virtual]

Compares equality between this object and the one passed.

**Parameters**

<i>f</i>	- the value to be compared to this one.
----------	---

**Returns**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Float** > (p. 1250).

**6.364.2.21** `virtual bool decaf::lang::Float::operator==( const float & f ) const` [inline, virtual]

Compares equality between this object and the one passed.

**Parameters**

<i>f</i>	- the value to be compared to this one.
----------	---

**Returns**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **float** > (p. 1250).

**6.364.2.22** `static float decaf::lang::Float::parseFloat ( const std::string & value ) throw ( exceptions::NumberFormatException )` [static]

Returns a new float initialized to the value represented by the specified string, as performed by the valueOf method of class **Float** (p. 1961).

**Parameters**

<i>value</i>	- the string to parse
--------------	-----------------------

**Returns**

a float parsed from the string

## Exceptions

<i>NumberFormatException</i>	
------------------------------	--

**6.364.2.23** `virtual short decaf::lang::Float::shortValue ( ) const` [inline, virtual]

Answers the short value which the receiver represents.

## Returns

short the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2920).

**6.364.2.24** `static std::string decaf::lang::Float::toHexString ( float value )` [static]

Returns a hexadecimal string representation of the float argument.

All characters mentioned below are ASCII characters.

\* If the argument is NaN, the result is the string "NaN". \* Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the string "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the string "0x0.0p0"; thus, negative zero produces the result "-0x0.0p0" and positive zero produces the result "0x0.0p0". o If m is a float value with a normalized representation, substrings are used to represent the significand and exponent fields. The significand is represented by the characters "0x1." followed by a lowercase hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed unless all the digits are zero, in which case a single zero is used. Next, the exponent is represented by "p" followed by a decimal string of the unbiased exponent as if produced by a call to **Integer.toString** (p. 2157) on the exponent value. o If m is a float value with a subnormal representation, the significand is represented by the characters "0x0." followed by a hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed. Next, the exponent is represented by "p-126". Note that there must be at least one nonzero digit in a subnormal significand.

## Parameters

<i>value</i>	- The float to convert to a string
--------------	------------------------------------

## Returns

the Hex formatted float string.

**6.364.2.25** `std::string decaf::lang::Float::toString ( ) const`

#### Returns

this **Float** (p. 1961) Object as a **String** (p. 3775) Representation

**6.364.2.26** `static std::string decaf::lang::Float::toString ( float value ) [static]`

Returns a string representation of the float argument.

All characters mentioned below are ASCII characters.

If the argument is NaN, the result is the string "NaN". Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m:

- o If m is infinity, it is represented by the characters "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity".
- o If m is zero, it is represented by the characters "0.0"; thus, negative zero produces the result "-0.0" and positive zero produces the result "0.0".
- o If m is greater than or equal to 10<sup>-3</sup> but less than 10<sup>7</sup>, then it is represented as the integer part of m, in decimal form with no leading zeroes, followed by '.', followed by one or more decimal digits representing the fractional part of m.
- o If m is less than 10<sup>-3</sup> or greater than or equal to 10<sup>7</sup>, then it is represented in so-called "computerized scientific notation." Let n be the unique integer such that 10<sup>n</sup> ≤ m < 10<sup>n+1</sup>; then let a be the mathematically exact quotient of m and 10<sup>n</sup> so that 1 ≤ a < 10. The magnitude is then represented as the integer part of a, as a single decimal digit, followed by '.', followed by decimal digits representing the fractional part of a, followed by the letter 'E', followed by a representation of n as a decimal integer, as produced by the method **Integer.toString(int)** (p. 2158).

#### Parameters

<i>value</i>	- The float to convert to a string
--------------	------------------------------------

#### Returns

the formatted float string.

**6.364.2.27** `static Float decaf::lang::Float::valueOf ( const std::string & value ) throw ( exceptions::NumberFormatException ) [static]`

Returns a **Float** (p. 1961) instance that wraps a primitive float which is parsed from the string value passed.

#### Parameters

<i>value</i>	- the string to parse
--------------	-----------------------

**Returns**

a new **Float** (p. 1961) instance wrapping the float parsed from value

**Exceptions**

<i>NumberFormatException</i>	on error.
------------------------------	-----------

**6.364.2.28** `static Float decaf::lang::Float::valueOf ( float value ) [static]`

Returns a **Float** (p. 1961) instance representing the specified float value.

**Parameters**

<i>value</i>	- float to wrap
--------------	-----------------

**Returns**

new **Float** (p. 1961) instance wrapping the primitive value

**6.364.3 Field Documentation**

**6.364.3.1** `const float decaf::lang::Float::MAX_VALUE [static]`

The maximum value that the primitive type can hold.

**6.364.3.2** `const float decaf::lang::Float::MIN_VALUE [static]`

The minimum value that the primitive type can hold.

**6.364.3.3** `const float decaf::lang::Float::NaN [static]`

Constant for the Not a **Number** (p. 2918) Value.

**6.364.3.4** `const float decaf::lang::Float::NEGATIVE_INFINITY [static]`

Constant for Negative Infinity.

**6.364.3.5** `const float decaf::lang::Float::POSITIVE_INFINITY [static]`

Constant for Positive Infinity.

**6.364.3.6** `const int decaf::lang::Float::SIZE = 32 [static]`

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Float.h`

## 6.365 decaf::internal::nio::FloatArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/FloatArrayBuffer.h>
```

Inheritance diagram for `decaf::internal::nio::FloatArrayBuffer`:

### Public Member Functions

- **FloatArrayBuffer** (int size, bool readOnly=false) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **FloatArrayBuffer** (p. 1974) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **FloatArrayBuffer** (float \*array, int size, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a **FloatArrayBuffer** (p. 1974) object that wraps the given array.*
- **FloatArrayBuffer** (const decaf::lang::Pointer< ByteArrayAdapter > &array, int offset, int capacity, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.*
- **FloatArrayBuffer** (const FloatArrayBuffer &other)  
*Create a **FloatArrayBuffer** (p. 1974) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*
- virtual ~**FloatArrayBuffer** ()
- virtual float \* **array** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )  
*Returns the float array that backs this buffer (optional operation).  
Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.  
Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.*  
**Returns**  
*the array that backs this **Buffer** (p. 936).*  
**Exceptions**



<b>ReadOnlyBufferException</b> (p. 3260)	if this <i>Buffer</i> (p. 936) is read only.
UnsupportedOperationException	if the underlying store has no array.

- virtual int **arrayOffset** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the *hasArray* method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

The offset into the backing array where index zero starts.

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3260)	if this <i>Buffer</i> (p. 936) is read only.
UnsupportedOperationException	if the underlying store has no array.

- virtual FloatBuffer \* **asReadOnlyBuffer** () const

Creates a new, read-only float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the *duplicate* method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns

The new, read-only float buffer which the caller then owns.

- virtual FloatBuffer & **compact** () throw ( decaf::nio::ReadOnlyBufferException )

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 941) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 940) - 1 is copied to index  $n = \text{limit}()$  (p. 940) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

**Returns**

a reference to this **FloatBuffer** (p. 1985).

**Exceptions**

<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only
---	-----------------------------

- virtual **FloatBuffer \* duplicate ()**

Creates a new float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns**

a new float **Buffer** (p. 936) which the caller owns.

- virtual float **get ()** throw ( decaf::nio::BufferUnderflowException )

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

**Returns**

the float at the current position.

**Exceptions**

<b>BufferUnderflowException</b> (p. 967)	if there no more data to return.
---	----------------------------------

- virtual float **get (int index)** const throw ( lang::exceptions::IndexOutOfBoundsException )

Absolute get method.

Reads the value at the given index.

**Parameters**

index	The index in the <b>Buffer</b> (p. 936) where the float is to be read
-------	---

**Returns**

the float that is located at the given index

**Exceptions**

<b>IndexOutOfBoundsException</b>	if index is not smaller than the buffer's limit
----------------------------------	---

- virtual bool **hasArray ()** const

Tells whether or not this buffer is backed by an accessible float array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns**

*true if, and only if, this buffer is backed by an array and is not read-only*

- virtual bool **isReadOnly** () const

*Tells whether or not this buffer is read-only.*

**Returns**

*true if, and only if, this buffer is read-only.*

- virtual FloatBuffer & **put** (float value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes the given floats into this buffer at the current position, and then increments the position.*

**Parameters**

value	<i>The floats value to be written.</i>
-------	--

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 964)	<i>if this buffer's current position is not smaller than its limit</i>
<b>ReadOnlyBufferException</b> (p. 3260)	<i>if this buffer is read-only</i>

- virtual FloatBuffer & **put** (int index, float value) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes the given floats into this buffer at the given index.*

**Parameters**

index	<i>The position in the <b>Buffer</b> (p. 936) to write the data.</i>
value	<i>The floats to write.</i>

**Returns**

*a reference to this buffer.*

**Exceptions**

IndexOutOfBoundsException	<i>if index greater than the buffer's limit minus the size of the type being written, or index is negative.</i>
<b>ReadOnlyBufferException</b> (p. 3260)	<i>if this buffer is read-only.</i>

- virtual FloatBuffer \* **slice** () const

*Creates a new **FloatBuffer** (p. 1985) whose content is a shared subsequence of this buffer's content.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers'*

*position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

#### **Returns**

*the newly create **FloatBuffer** (p. 1985) which the caller owns.*

## **Protected Member Functions**

- virtual void **setReadOnly** (bool value)

*Sets this **FloatArrayBuffer** (p. 1974) as Read-Only.*

## **6.365.1 Constructor & Destructor Documentation**

**6.365.1.1** `decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer ( int size, bool readOnly = false ) throw ( decaf::lang::exceptions::IllegalArgumentException )`

Creates a **FloatArrayBuffer** (p. 1974) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

### **Parameters**

<i>size</i>	The size of the array, this is the limit we read and write to.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

### **Exceptions**

<i>IllegalArgumentException</i>	if the capacity value is negative.
---------------------------------	------------------------------------

**6.365.1.2** `decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer ( float * array, int size, int offset, int length, bool readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a **FloatArrayBuffer** (p. 1974) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

### **Parameters**

<i>array</i>	The actual array to wrap.
<i>size</i>	The size of the given array.
<i>offset</i>	The position that is this buffers start position.

<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

**Exceptions**

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if offset is greater than array capacity.

**6.365.1.3** `decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer ( const decaf::lang::Pointer< ByteArrayAdapter > & array, int offset, int capacity, bool readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **FloatArrayBuffer** (p. 1974) will be that of the remaining capacity of the passed buffer.

**Parameters**

<i>array</i>	The ByteArrayAdapter to wrap.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

**Exceptions**

<i>NullPointerException</i>	if array is NULL
<i>IndexOutOfBoundsException</i>	if offset + length is greater than array size.

**6.365.1.4** `decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer ( const FloatArrayBuffer & other )`

Create a **FloatArrayBuffer** (p. 1974) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.

**Parameters**

<i>other</i>	The <b>FloatArrayBuffer</b> (p. 1974) this one is to mirror.
--------------	--

6.365.1.5 `virtual decaf::internal::nio::FloatArrayBuffer::~FloatArrayBuffer ( )` [virtual]

## 6.365.2 Member Function Documentation

6.365.2.1 `virtual float* decaf::internal::nio::FloatArrayBuffer::array ( )` throw (`decaf::lang::exceptions::UnsupportedOperationException`, `decaf::nio::ReadOnlyBufferException`) [virtual]

Returns the float array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

the array that backs this **Buffer** (p. 936).

### Exceptions

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<i><b>UnsupportedOperationException</b></i>	if the underlying store has no array.

Implements `decaf::nio::FloatBuffer` (p. 1989).

6.365.2.2 `virtual int decaf::internal::nio::FloatArrayBuffer::arrayOffset ( )` throw (`decaf::lang::exceptions::UnsupportedOperationException`, `decaf::nio::ReadOnlyBufferException`) [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

The offset into the backing array where index zero starts.

### Exceptions

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<i><b>UnsupportedOperationException</b></i>	if the underlying store has no array.

Implements **decaf::nio::FloatBuffer** (p. 1989).

**6.365.2.3** `virtual FloatBuffer* decaf::internal::nio::FloatArrayBuffer::asReadOnlyBuffer ( )`  
`const [virtual]`

Creates a new, read-only float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns

The new, read-only float buffer which the caller then owns.

Implements **decaf::nio::FloatBuffer** (p. 1990).

**6.365.2.4** `virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::compact ( ) throw (`  
`decaf::nio::ReadOnlyBufferException) [virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 941) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 940) - 1 is copied to index  $n = \text{limit}()$  (p. 940) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

#### Returns

a reference to this **FloatBuffer** (p. 1985).

#### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only
--	-----------------------------

Implements **decaf::nio::FloatBuffer** (p. 1990).

**6.365.2.5** `virtual FloatBuffer* decaf::internal::nio::FloatArrayBuffer::duplicate ( )`  
`[virtual]`

Creates a new float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

a new float **Buffer** (p. 936) which the caller owns.

Implements **decaf::nio::FloatBuffer** (p. 1991).

**6.365.2.6** `virtual float decaf::internal::nio::FloatArrayBuffer::get ( ) throw (`  
`decaf::nio::BufferUnderflowException ) [virtual]`

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

### Returns

the float at the current position.

### Exceptions

<i><b>BufferUnderflowException</b></i> (p. 967)	if there no more data to return.
--	----------------------------------

Implements **decaf::nio::FloatBuffer** (p. 1993).

**6.365.2.7** `virtual float decaf::internal::nio::FloatArrayBuffer::get ( int index ) const throw (`  
`lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Absolute get method.

Reads the value at the given index.

### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the float is to be read
--------------	---

### Returns

the float that is located at the given index



### Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit
----------------------------------	---

Implements **decaf::nio::FloatBuffer** (p. 1991).

**6.365.2.8** `virtual bool decaf::internal::nio::FloatArrayBuffer::hasArray ( ) const [inline, virtual]`

Tells whether or not this buffer is backed by an accessible float array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

### Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::FloatBuffer** (p. 1993).

**6.365.2.9** `virtual bool decaf::internal::nio::FloatArrayBuffer::isReadOnly ( ) const [inline, virtual]`

Tells whether or not this buffer is read-only.

### Returns

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 940).

**6.365.2.10** `virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::put ( int index, float value ) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Writes the given floats into this buffer at the given index.

### Parameters

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data.
<i>value</i>	The floats to write.

### Returns

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only.

Implements **decaf::nio::FloatBuffer** (p. 1994).

**6.365.2.11** `virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::put ( float value ) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )` [virtual]

Writes the given floats into this buffer at the current position, and then increments the position.

**Parameters**

<i>value</i>	The floats value to be written.
--------------	---------------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b>BufferOverflowException</b> (p. 964)	if this buffer's current position is not smaller than its limit
<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only

Implements **decaf::nio::FloatBuffer** (p. 1993).

**6.365.2.12** `virtual void decaf::internal::nio::FloatArrayBuffer::setReadOnly ( bool value )` [inline, protected, virtual]

Sets this **FloatArrayBuffer** (p. 1974) as Read-Only.

**Parameters**

<i>value</i>	Boolean value, true if this buffer is to be read-only, false otherwise.
--------------	---

**6.365.2.13** `virtual FloatBuffer* decaf::internal::nio::FloatArrayBuffer::slice ( ) const`  
`[virtual]`

Creates a new **FloatBuffer** (p. 1985) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the newly create **FloatBuffer** (p. 1985) which the caller owns.

Implements **decaf::nio::FloatBuffer** (p. 1996).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/FloatArrayBuffer.h`

## 6.366 decaf::nio::FloatBuffer Class Reference

This class defines four categories of operations upon float buffers:

```
#include <src/main/decaf/nio/FloatBuffer.h>
```

Inheritance diagram for `decaf::nio::FloatBuffer`:

### Public Member Functions

- `virtual ~FloatBuffer ()`
- `virtual std::string toString () const`
- `virtual float * array ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )`  
*Returns the float array that backs this buffer (optional operation).*
- `virtual int arrayOffset ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )`  
*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*
- `virtual FloatBuffer * asReadOnlyBuffer () const =0`  
*Creates a new, read-only float buffer that shares this buffer's content.*

- virtual **FloatBuffer** & **compact** ()=0 throw ( ReadOnlyBufferException )  
*Compacts this buffer.*
- virtual **FloatBuffer** \* **duplicate** ()=0  
*Creates a new float buffer that shares this buffer's content.*
- virtual float **get** ()=0 throw ( BufferUnderflowException )  
*Relative get method.*
- virtual float **get** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Absolute get method.*
- **FloatBuffer** & **get** (std::vector< float > buffer) throw ( BufferUnderflowException )  
*Relative bulk get method.*
- **FloatBuffer** & **get** (float \*buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Relative bulk get method.*
- virtual bool **hasArray** () const =0  
*Tells whether or not this buffer is backed by an accessible float array.*
- **FloatBuffer** & **put** (**FloatBuffer** &src) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )  
*This method transfers the floats remaining in the given source buffer into this buffer.*
- **FloatBuffer** & **put** (const float \*buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*This method transfers floats into this buffer from the given source array.*
- **FloatBuffer** & **put** (std::vector< float > &buffer) throw ( BufferOverflowException, ReadOnlyBufferException )  
*This method transfers the entire content of the given source floats array into this buffer.*
- virtual **FloatBuffer** & **put** (float value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )  
*Writes the given floats into this buffer at the current position, and then increments the position.*
- virtual **FloatBuffer** & **put** (int index, float value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )

*Writes the given floats into this buffer at the given index.*

- virtual **FloatBuffer** \* **slice** () const =0  
*Creates a new **FloatBuffer** (p. 1985) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (const **FloatBuffer** &value) const
- virtual bool **equals** (const **FloatBuffer** &value) const
- virtual bool **operator==** (const **FloatBuffer** &value) const
- virtual bool **operator<** (const **FloatBuffer** &value) const

### Static Public Member Functions

- static **FloatBuffer** \* **allocate** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Allocates a new Double buffer.*
- static **FloatBuffer** \* **wrap** (float \*array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Wraps the passed buffer with a new **FloatBuffer** (p. 1985).*
- static **FloatBuffer** \* **wrap** (std::vector< float > &buffer)  
*Wraps the passed STL float Vector in a **FloatBuffer** (p. 1985).*

### Protected Member Functions

- **FloatBuffer** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **FloatBuffer** (p. 1985) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

### 6.366.1 Detailed Description

This class defines four categories of operations upon float buffers: o Absolute and relative get and put methods that read and write single floats; o Relative bulk get methods

that transfer contiguous sequences of floats from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of floats from a float array or some other float buffer into this buffer o Methods for compacting, duplicating, and slicing a float buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing float array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

### 6.366.2 Constructor & Destructor Documentation

**6.366.2.1** `decaf::nio::FloatBuffer::FloatBuffer ( int capacity ) throw ( decaf::lang::exceptions::IllegalArgumentException )` [protected]

Creates a **FloatBuffer** (p. 1985) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

#### Parameters

<i>capacity</i>	The size and limit of the <b>Buffer</b> (p. 936) in floats.
-----------------	---

#### Exceptions

<i>IllegalArgumentException</i>	if capacity is negative.
---------------------------------	--------------------------

**6.366.2.2** `virtual decaf::nio::FloatBuffer::~~FloatBuffer ( )` [inline, virtual]

### 6.366.3 Member Function Documentation

**6.366.3.1** `static FloatBuffer* decaf::nio::FloatBuffer::allocate ( int capacity ) throw ( decaf::lang::exceptions::IllegalArgumentException )` [static]

Allocates a new Double buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

#### Parameters

<i>capacity</i>	The size of the Double buffer in floats.
-----------------	--

#### Returns

the **FloatBuffer** (p. 1985) that was allocated, caller owns.

**6.366.3.2** virtual float\* decaf::nio::FloatBuffer::array ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]

Returns the float array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

the array that backs this **Buffer** (p. 936).

#### Exceptions

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1980).

**6.366.3.3** virtual int decaf::nio::FloatBuffer::arrayOffset ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

The offset into the backing array where index zero starts.

#### Exceptions

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1980).

**6.366.3.4** `virtual FloatBuffer* decaf::nio::FloatBuffer::asReadOnlyBuffer ( ) const` [pure virtual]

Creates a new, read-only float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns

The new, read-only float buffer which the caller then owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1981).

**6.366.3.5** `virtual FloatBuffer& decaf::nio::FloatBuffer::compact ( ) throw ( ReadOnlyBufferException )` [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 941) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 940) - 1 is copied to index  $n = \text{limit}()$  (p. 940) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

#### Returns

a reference to this **FloatBuffer** (p. 1985).

#### Exceptions

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this buffer is read-only
--	-----------------------------

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1981).



**6.366.3.6** `virtual int decaf::nio::FloatBuffer::compareTo ( const FloatBuffer & value ) const`  
[virtual]

**6.366.3.7** `virtual FloatBuffer* decaf::nio::FloatBuffer::duplicate ( )` [pure virtual]

Creates a new float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

a new float **Buffer** (p. 936) which the caller owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1982).

**6.366.3.8** `virtual bool decaf::nio::FloatBuffer::equals ( const FloatBuffer & value ) const`  
[virtual]

**6.366.3.9** `FloatBuffer& decaf::nio::FloatBuffer::get ( std::vector< float > buffer ) throw ( BufferUnderflowException )`

Relative bulk get method.

This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize( N )` before calling this get method.

#### Returns

a reference to this **Buffer** (p. 936).

#### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 967)	if there are fewer than length floats remaining in this buffer
--	--

**6.366.3.10** `virtual float decaf::nio::FloatBuffer::get ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [pure virtual]

Absolute get method.

Reads the value at the given index.

**Parameters**

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the float is to be read
--------------	---

**Returns**

the float that is located at the given index

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit
----------------------------------	---

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1982).

**6.366.3.11 FloatBuffer& decaf::nio::FloatBuffer::get ( float \* *buffer*, int *size*, int *offset*, int *length* ) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )**

Relative bulk get method.

This method transfers floats from this buffer into the given destination array. If there are fewer floats remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 941), then no bytes are transferred and a **BufferUnderflowException** (p. 967) is thrown.

Otherwise, this method copies `length` floats from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

**Parameters**

<i>buffer</i>	The pointer to an allocated buffer to fill.
<i>size</i>	The size of the passed in buffer.
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The amount of data to put in the passed buffer.

**Returns**

a reference to this **Buffer** (p. 936).

**Exceptions**

<i>BufferUnderflowException</i> (p. 967)	if there are fewer than <code>length</code> floats remaining in this buffer
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of <code>size</code> , <code>offset</code> , or <code>length</code> are not met.

**6.366.3.12** `virtual float decaf::nio::FloatBuffer::get ( ) throw ( BufferUnderflowException )`  
[pure virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

#### Returns

the float at the current position.

#### Exceptions

<i><b>BufferUnderflowException</b></i> (p. 967)	if there no more data to return.
--	----------------------------------

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1982).

**6.366.3.13** `virtual bool decaf::nio::FloatBuffer::hasArray ( ) const` [pure virtual]

Tells whether or not this buffer is backed by an accessible float array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

#### Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1983).

**6.366.3.14** `virtual bool decaf::nio::FloatBuffer::operator< ( const FloatBuffer & value ) const`  
[virtual]

**6.366.3.15** `virtual bool decaf::nio::FloatBuffer::operator== ( const FloatBuffer & value ) const`  
[virtual]

**6.366.3.16** `virtual FloatBuffer& decaf::nio::FloatBuffer::put ( float value ) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes the given floats into this buffer at the current position, and then increments the position.

#### Parameters

<i>value</i>	The floats value to be written.
--------------	---------------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 964)	if this buffer's current position is not smaller than its limit
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1984).

**6.366.3.17** **virtual FloatBuffer& decaf::nio::FloatBuffer::put ( int *index*, float *value* )**  
**throw ( decaf::lang::exceptions::IndexOutOfBoundsException,**  
**ReadOnlyBufferException )** [pure virtual]

Writes the given floats into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data.
<i>value</i>	The floats to write.

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>IndexOutOfBoundsException</i></b>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1983).

**6.366.3.18** **FloatBuffer& decaf::nio::FloatBuffer::put ( const float \* *buffer*, int *size*, int *offset*, int *length* )** throw ( **BufferOverflowException**, **ReadOnlyBufferException**, **decaf::lang::exceptions::IndexOutOfBoundsException**, **decaf::lang::exceptions::NullPointerException** )

This method transfers floats into this buffer from the given source array.

If there are more floats to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 941), then no floats are transferred and a **BufferOverflowException** (p. 964) is thrown.

Otherwise, this method copies length bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by length.

#### Parameters

<i>buffer</i>	The array from which floats are to be read.
<i>size</i>	The size of the passed in buffer.
<i>offset</i>	The offset within the array of the first float to be read.
<i>length</i>	The number of floats to be read from the given array.

#### Returns

a reference to this buffer.

#### Exceptions

<b><i>BufferOverflowException</i></b> (p. 964)	if there is insufficient space in this buffer
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

#### 6.366.3.19 FloatBuffer& decaf::nio::FloatBuffer::put ( std::vector< float > & buffer ) throw ( BufferOverflowException, ReadOnlyBufferException )

This method transfers the entire content of the given source floats array into this buffer. This is the same as calling put( &buffer[0], 0, buffer.size()).

#### Parameters

<i>buffer</i>	The buffer whose contents are copied to this <b>FloatBuffer</b> (p. 1985)
---------------	---

#### Returns

a reference to this buffer.

#### Exceptions

<b><i>BufferOverflowException</i></b> (p. 964)	if there is insufficient space in this buffer
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only

**6.366.3.20** `FloatBuffer& decaf::nio::FloatBuffer::put ( FloatBuffer & src )`  
`throw ( BufferOverflowException, ReadOnlyBufferException,`  
`decaf::lang::exceptions::IllegalArgumentException )`

This method transfers the floats remaining in the given source buffer into this buffer.

If there are more floats remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 941), then no floats are transferred and a **BufferOverflowException** (p. 964) is thrown.

Otherwise, this method copies `n = src.remaining()` floats from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

#### Parameters

<i>src</i>	The buffer to take floats from an place in this one.
------------	--

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 964)	if there is insufficient space in this buffer for the remaining floats in the source buffer
<b>IllegalArgumentException</b> <i>exception</i>	if the source buffer is this buffer.
<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only.

**6.366.3.21** `virtual FloatBuffer* decaf::nio::FloatBuffer::slice ( ) const` [pure virtual]

Creates a new **FloatBuffer** (p. 1985) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the newly create **FloatBuffer** (p. 1985) which the caller owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1985).

**6.366.3.22** `virtual std::string decaf::nio::FloatBuffer::toString ( ) const` `[virtual]`

#### Returns

a std::string describing this object

**6.366.3.23** `static FloatBuffer* decaf::nio::FloatBuffer::wrap ( float * array, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` `[static]`

Wraps the passed buffer with a new **FloatBuffer** (p. 1985).

The new buffer will be backed by the given float array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>array</i>	The array that will back the new buffer.
<i>size</i>	The size of the array that was passed in.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

#### Returns

a new **FloatBuffer** (p. 1985) that is backed by buffer, caller owns.

#### Exceptions

<i>NullPointerException</i>	if the array pointer is NULL.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

**6.366.3.24** `static FloatBuffer* decaf::nio::FloatBuffer::wrap ( std::vector< float > & buffer )` `[static]`

Wraps the passed STL float Vector in a **FloatBuffer** (p. 1985).

The new buffer will be backed by the given float array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	- The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).
---------------	--

**Returns**

a new **FloatBuffer** (p. 1985) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**FloatBuffer.h**

**6.367 decaf::io::Flushable Class Reference**

A **Flushable** (p. 1998) is a destination of data that can be flushed.

```
#include <src/main/decaf/io/Flushable.h>
```

Inheritance diagram for decaf::io::Flushable:

**Public Member Functions**

- virtual **~Flushable** ()
- virtual void **flush** ()=0 throw ( decaf::io::IOException )  
*Flushes this stream by writing any buffered output to the underlying stream.*

**6.367.1 Detailed Description**

A **Flushable** (p. 1998) is a destination of data that can be flushed. The flush method is invoked to write any buffered output to the underlying stream.

**Since**

1.0

**6.367.2 Constructor & Destructor Documentation**

**6.367.2.1** virtual decaf::io::Flushable::~Flushable ( ) [inline, virtual]

**6.367.3 Member Function Documentation**

**6.367.3.1** virtual void decaf::io::Flushable::flush ( ) throw ( decaf::io::IOException )  
 [pure virtual]

Flushes this stream by writing any buffered output to the underlying stream.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
--	-------------------------



Implemented in **decaf::internal::io::StandardErrorOutputStream** (p. 3688), **decaf::internal::io::StandardOutputStream** (p. 3691), **decaf::io::BufferedOutputStream** (p. 951), **decaf::io::FilterOutputStream** (p. 1960), **decaf::io::OutputStream** (p. 2994), and **decaf::io::OutputStreamWriter** (p. 3001).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**Flushable.h**

## 6.368 activemq::commands::FlushCommand Class Reference

```
#include <src/main/activemq/commands/FlushCommand.h>
```

Inheritance diagram for activemq::commands::FlushCommand:

### Public Member Functions

- **FlushCommand** ()
- virtual **~FlushCommand** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **FlushCommand \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

**Static Public Attributes**

- static const unsigned char **ID\_FLUSHCOMMAND** = 15

**6.368.1 Constructor & Destructor Documentation**

**6.368.1.1** `activemq::commands::FlushCommand::FlushCommand ( )`

**6.368.1.2** `virtual activemq::commands::FlushCommand::~~FlushCommand ( )`  
[virtual]

**6.368.2 Member Function Documentation**

**6.368.2.1** `virtual FlushCommand* activemq::commands::FlushCommand::cloneDataStructure ( ) const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

**Returns**

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

**6.368.2.2** `virtual void activemq::commands::FlushCommand::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

**Parameters**

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

**6.368.2.3** `virtual bool activemq::commands::FlushCommand::equals ( const DataStructure * value ) const` [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

**6.368.2.4** `virtual unsigned char activemq::commands::FlushCommand::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

**6.368.2.5** `virtual std::string activemq::commands::FlushCommand::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

**6.368.2.6** `virtual Pointer<Command> activemq::commands::FlushCommand::visit ( activemq::state::CommandVisitor * visitor ) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

### 6.368.3 Field Documentation

**6.368.3.1** `const unsigned char activemq::commands::FlushCommand::ID_FLUSHCOMMAND = 15 [static]`

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**FlushCommand.h**

## 6.369 activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2002).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/FlushCommandMar
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller:

### Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

6.369

activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller

Class Reference

2005

### 6.369.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2002).  
NOTE!: This file is auto generated - do not modify! if you need to make a change,  
please see the Java Classes in the activemq-openwire-generator module

### 6.369.2 Constructor & Destructor Documentation

6.369.2.1 **activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::FlushCommandMarshaller**  
( ) [inline]

6.369.2.2 **virtual activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::~~FlushCommandMarshaller**  
( ) [inline, virtual]

### 6.369.3 Member Function Documentation

6.369.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::createObject** (  
) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.369.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.369.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure**  
\* **dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw (  
**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 801).

**6.369.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 802).

**6.369.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

6.369

**activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller****Class Reference****2007****Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 803).

**6.369.3.6** virtual void **activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::tightMarshal2**  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
 \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*,  
**utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** )  
 [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 804).

**6.369.3.7** virtual void **activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::tightUnmarshal**  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream**  
 \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 806).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**FlushCommandMarshaller.h**

## 6.370 **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2006).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMar
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller**:

**Public Member Functions**

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*



## 6.370

**activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller**

### Class Reference

2009

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.370.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2006).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.370.2 Constructor & Destructor Documentation

6.370.2.1 **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::FlushCommandMarshaller**  
( ) [inline]

6.370.2.2 **virtual activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::~FlushCommandMarshaller**  
( ) [inline, virtual]

### 6.370.3 Member Function Documentation

6.370.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::createObject** (  
) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.370.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.370.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 808).

**6.370.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 809).

6.370

activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller

Class Reference

2011

```
6.370.3.5 virtual int activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 810).

```
6.370.3.6 virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 811).

```
6.370.3.7 virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 813).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**FlushCommandMarshaller.h**

## 6.371 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2010).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMar
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller**:

#### Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*

## 6.371

**activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller**

### Class Reference

2013

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.371.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p.2010).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.371.2 Constructor & Destructor Documentation

6.371.2.1 **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::FlushCommandMarshaller**  
( ) [inline]

6.371.2.2 **virtual activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::~~FlushCommandMarshaller**  
( ) [inline, virtual]

### 6.371.3 Member Function Documentation

6.371.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::createObject** (  
) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.371.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::getDataStructureType  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.371.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::looseMarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 772).

**6.371.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::looseUnmarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (  
decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

## 6.371

**activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller**

### Class Reference

2015

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 774).

**6.371.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 775).

**6.371.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 776).

**6.371.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h`

## 6.372 **activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2014).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMar
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller**:



- **FlushCommandMarshaller ()**
- virtual **~FlushCommandMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.372.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2014).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.372.2 Constructor & Destructor Documentation

6.372.2.1 `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::FlushCommandMarshaller ( ) [inline]`

6.372.2.2 `virtual activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::~~FlushCommandMarshaller ( ) [inline, virtual]`

## 6.372.3 Member Function Documentation

6.372.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.372.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.372.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

6.372

**activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller****Class Reference****2019****Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 780).

**6.372.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 781).

**6.372.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 782).

```
6.372.3.6 virtual void activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 783).

```
6.372.3.7 virtual void activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 784).

The documentation for this class was generated from the following file:

6.373

**activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller**

**Class Reference**

2021

- `src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h`

## 6.373 **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller** **Class Reference**

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2019).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller`:

### Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.373.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2019).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.373.2 Constructor & Destructor Documentation

6.373.2.1 **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::FlushCommandMarshaller**  
 ( ) [inline]

6.373.2.2 **virtual activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::~~FlushCommandMarshaller**  
 ( ) [inline, virtual]

### 6.373.3 Member Function Documentation

6.373.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::createObject (**  
**) const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.373.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::getDataStructureType**  
**( ) const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.373.3.3 **virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure**  
**\* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw (**  
**decaf::io::IOException )** [virtual]

Write a object instance to data output stream.

### 6.373

**activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller**

#### Class Reference

2023

##### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

##### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 787).

**6.373.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

##### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

##### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 788).

**6.373.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

##### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 789).

```
6.373.3.6 virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 790).

```
6.373.3.7 virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.



6.374

**activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller**

**Class Reference**

2025

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 792).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**FlushCommandMarshaller.h**

## 6.374 **activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2023).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller**:

### Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.374.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2023).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.374.2 Constructor & Destructor Documentation

- 6.374.2.1 **activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::FlushCommandMarshaller**  
 ( ) [inline]
- 6.374.2.2 **virtual activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::~~FlushCommandMarshaller**  
 ( ) [inline, virtual]

### 6.374.3 Member Function Documentation

- 6.374.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::createObject** ( ) **const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

- 6.374.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::getDataStructureType**  
 ( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

6.374

**activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller**

**Class Reference**

2027

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.374.3.3** virtual void **activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::looseMarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
\* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw (  
**decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller**  
(p. 794).

**6.374.3.4** virtual void **activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::looseUnmarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
\* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw (  
**decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller**  
(p. 795).

```

6.374.3.5  virtual int activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 796).

```

6.374.3.6  virtual void activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 797).

6.374.3.7 virtual void activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**FlushCommandMarshaller.h**

## 6.375 decaf::util::logging::Formatter Class Reference

A **Formatter** (p. 2027) provides support for formatting LogRecords.

```
#include <src/main/decaf/util/logging/Formatter.h>
```

Inheritance diagram for decaf::util::logging::Formatter:

#### Public Member Functions

- virtual ~**Formatter** ()
- virtual std::string **format** (const **LogRecord** &record) const =0  
*Format the given log record and return the formatted string.*
- virtual std::string **formatMessage** (const **LogRecord** &record) const  
*Format the message string from a log record.*
- virtual std::string **getHead** (const **Handler** \*handler DECAF\_UNUSED)  
*Return the header string for a set of formatted records.*

- virtual std::string **getTail** (const **Handler** \*handler DECAF\_UNUSED)

*Return the tail string for a set of formatted records.*

### 6.375.1 Detailed Description

A **Formatter** (p. 2027) provides support for formatting LogRecords. Typically each logging **Handler** (p. 2042) will have a **Formatter** (p. 2027) associated with it. The **Formatter** (p. 2027) takes a **LogRecord** (p. 2487) and converts it to a string.

Some formatters (such as the **XMLFormatter** (p. 4172)) need to wrap head and tail strings around a set of formatted records. The `getHeader` and `getTail` methods can be used to obtain these strings.

### 6.375.2 Constructor & Destructor Documentation

6.375.2.1 virtual decaf::util::logging::Formatter::~Formatter ( ) [inline, virtual]

### 6.375.3 Member Function Documentation

6.375.3.1 virtual std::string decaf::util::logging::Formatter::format ( const **LogRecord** & *record* ) const [pure virtual]

Format the given log record and return the formatted string.

#### Parameters

<i>record</i>	The Log Record to Format
---------------	--------------------------

#### Returns

the formatted record.

Implemented in **decaf::util::logging::SimpleFormatter** (p. 3605), and **decaf::util::logging::XMLFormatter** (p. 4173).

6.375.3.2 virtual std::string decaf::util::logging::Formatter::formatMessage ( const **LogRecord** & *record* ) const [virtual]

Format the message string from a log record.

#### Parameters

<i>record</i>	The Log Record to Format
---------------	--------------------------

#### Returns

the formatted message

6.375.3.3 virtual std::string decaf::util::logging::Formatter::getHead ( const Handler \*handler  
**DECAF\_UNUSED** ) [inline, virtual]

Return the header string for a set of formatted records.

In the default implementation this method should return empty string.

#### Parameters

<i>handler</i>	The target handler, can be NULL.
----------------	----------------------------------

#### Returns

the head string.

6.375.3.4 virtual std::string decaf::util::logging::Formatter::getTail ( const Handler \*handler  
**DECAF\_UNUSED** ) [inline, virtual]

Return the tail string for a set of formatted records.

In the default implementation this method should return empty string

#### Parameters

<i>handler</i>	the target handler, can be null
----------------	---------------------------------

#### Returns

the tail string

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Formatter.h**

## 6.376 decaf::util::concurrent::Future< V > Class Template Reference

A **Future** (p. 2029) represents the result of an asynchronous computation.

```
#include <src/main/decaf/util/concurrent/Future.h>
```

#### Public Member Functions

- virtual ~**Future** ()
- bool **cancel** (bool mayInterruptIfRunning)=0  
*Attempts to cancel execution of this task.*
- bool **isCancelled** () const =0

*Returns true if this task was canceled before it completed normally.*

- **bool isDone () const** =0

*Returns true if this task completed.*

- **V get ()=0** throw ( CancellationException, InterruptedException, ExecutionException )

*Waits if necessary for the computation to complete, and then retrieves its result.*

- **V get (long long timeout, TimeUnit unit)=0** throw ( InterruptedException, ExecutionException, TimeoutException )

*Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.*

### 6.376.1 Detailed Description

**template<typename V> class decaf::util::concurrent::Future< V >**

A **Future** (p. 2029) represents the result of an asynchronous computation. Methods are provided to check if the computation is complete, to wait for its completion, and to retrieve the result of the computation. The result can only be retrieved using method `get` when the computation has completed, blocking if necessary until it is ready. Cancellation is performed by the `cancel` method. Additional methods are provided to determine if the task completed normally or was canceled. Once a computation has completed, the computation cannot be canceled. If you would like to use a **Future** (p. 2029) for the sake of cancellability but not provide a usable result, you can declare types of the form `Future<void*>` and return null as a result of the underlying task.

### 6.376.2 Constructor & Destructor Documentation

**6.376.2.1** **template<typename V > virtual decaf::util::concurrent::Future< V >::~~Future ( )** [inline, virtual]

### 6.376.3 Member Function Documentation

**6.376.3.1** **template<typename V > bool decaf::util::concurrent::Future< V >::cancel ( bool mayInterruptIfRunning )** [pure virtual]

Attempts to cancel execution of this task.

This attempt will fail if the task has already completed, has already been canceled, or could not be canceled for some other reason. If successful, and this task has not started when `cancel` is called, this task should never run. If the task has already started, then the `mayInterruptIfRunning` parameter determines whether the thread executing this task should be interrupted in an attempt to stop the task.



After this method returns, subsequent calls to **isDone()** (p. 2032) will always return true. Subsequent calls to **isCancelled()** (p. 2032) will always return true if this method returned true.

#### Parameters

<i>mayInterruptIfRunning</i>	- true if the thread executing this task should be interrupted; otherwise, in-progress tasks are allowed to complete.
------------------------------	---

#### Returns

false if the task could not be canceled, typically because it has already completed normally; true otherwise

**6.376.3.2** `template<typename V> V decaf::util::concurrent::Future< V >::get  
( long long timeout, TimeUnit unit ) throw ( InterruptedException,  
ExecutionException, TimeoutException ) [pure virtual]`

Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.

#### Parameters

<i>timeout</i>	- the maximum time to wait
<i>unit</i>	- the time unit of the timeout argument

#### Returns

the computed result

#### Exceptions

<b>CancellationException</b> (p. 1110)	- if the computation was canceled
<b>ExecutionException</b> (p. 1923)	- if the computation threw an exception
<b>InterruptedException</b>	- if the current thread was interrupted while waiting
<b>TimeoutException</b> (p. 3899)	- if the wait timed out

**6.376.3.3** `template<typename V> V decaf::util::concurrent::Future< V >::get ( )  
throw ( CancellationException, InterruptedException, ExecutionException )  
[pure virtual]`

Waits if necessary for the computation to complete, and then retrieves its result.

**Returns**

the computed result.

**Exceptions**

<i><b>CancellationException</b></i> (p. 1110)	- if the computation was canceled
<i><b>ExecutionException</b></i> (p. 1923)	- if the computation threw an exception
<i><b>InterruptedException</b></i>	- if the current thread was interrupted while waiting

**6.376.3.4** `template<typename V > bool decaf::util::concurrent::Future< V >::isCancelled ( ) const [pure virtual]`

Returns true if this task was canceled before it completed normally.

**Returns**

true if this task was canceled before it completed

**6.376.3.5** `template<typename V > bool decaf::util::concurrent::Future< V >::isDone ( ) const [pure virtual]`

Returns true if this task completed.

Completion may be due to normal termination, an exception, or cancellation -- in all of these cases, this method will return true.

**Returns**

true if this task completed

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Future.h`

## 6.377 activemq::transport::correlator::FutureResponse Class Reference

A container that holds a response object.

```
#include <src/main/activemq/transport/correlator/FutureResponse.h>
```

## Public Member Functions

- **FutureResponse** ()
- virtual **~FutureResponse** ()
- virtual const **Pointer**< **Response** > & **getResponse** () const
- virtual **Pointer**< **Response** > & **getResponse** (unsigned int timeout) const

*Getters for the response property.*

- virtual **Pointer**< **Response** > & **getResponse** ()
- virtual void **setResponse** (const **Pointer**< **Response** > &response)

*Getters for the response property.*

- virtual **Pointer**< **Response** > & **getResponse** (unsigned int timeout)
- virtual void **setResponse** (const **Pointer**< **Response** > &response)

*Setter for the response property.*

### 6.377.1 Detailed Description

A container that holds a response object. Callers of the `getResponse` method will block until a response has been receive unless they call the `getRepsonse` that takes a timeout.

### 6.377.2 Constructor & Destructor Documentation

**6.377.2.1** `activemq::transport::correlator::FutureResponse::FutureResponse ( )`  
[inline]

**6.377.2.2** `virtual activemq::transport::correlator::FutureResponse::~~FutureResponse ( )`  
[inline, virtual]

### 6.377.3 Member Function Documentation

**6.377.3.1** `virtual const Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse ( )`  
const [inline, virtual]

Getters for the response property.

Infinite Wait.

#### Returns

the response object for the request

**6.377.3.2** `virtual Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse ( )`  
`[inline, virtual]`

**6.377.3.3** `virtual Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse ( unsigned int timeout )` `[inline, virtual]`

**6.377.3.4** `virtual const Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse ( unsigned int timeout ) const` `[inline, virtual]`

Getters for the response property.

Timed Wait.

#### Parameters

<i>timeout</i>	- time to wait in milliseconds
----------------	--------------------------------

#### Returns

the response object for the request

**6.377.3.5** `virtual void activemq::transport::correlator::FutureResponse::setResponse ( const Pointer< Response > & response )` `[inline, virtual]`

Setter for the response property.

#### Parameters

<i>response</i>	the response object for the request.
-----------------	--------------------------------------

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/correlator/FutureResponse.h`

## 6.378 decaf::security::GeneralSecurityException Class Reference

```
#include <src/main/decaf/security/GeneralSecurityException.h>
```

Inheritance diagram for `decaf::security::GeneralSecurityException`:

#### Public Member Functions

- `GeneralSecurityException () throw ()`

*Default Constructor.*

- **GeneralSecurityException** (const **decaf::lang::Exception** &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **GeneralSecurityException** (const **GeneralSecurityException** &ex) throw ()  
*Copy Constructor.*
- **GeneralSecurityException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **GeneralSecurityException** (const std::exception \*cause) throw ()  
*Constructor.*
- **GeneralSecurityException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **GeneralSecurityException \* clone** () const  
*Clones this exception.*
- virtual ~**GeneralSecurityException** () throw ()

## 6.378.1 Constructor & Destructor Documentation

**6.378.1.1** decaf::security::GeneralSecurityException::GeneralSecurityException ( ) throw ()  
[inline]

Default Constructor.

**6.378.1.2** decaf::security::GeneralSecurityException::GeneralSecurityException ( const decaf::lang::Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.378.1.3** decaf::security::GeneralSecurityException::GeneralSecurityException ( const GeneralSecurityException & ex ) throw () [inline]

Copy Constructor.

**Parameters**

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.378.1.4** `decaf::security::GeneralSecurityException::GeneralSecurityException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.  
Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.378.1.5** `decaf::security::GeneralSecurityException::GeneralSecurityException ( const std::exception * cause ) throw () [inline]`

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.378.1.6** `decaf::security::GeneralSecurityException::GeneralSecurityException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.  
Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
<i>...</i>	list of primitives that are formatted into the message

## 6.379 decaf::internal::util::GenericResource< T > Class Template Reference 2039

6.378.1.7 virtual decaf::security::GeneralSecurityException::~~GeneralSecurityException ( )  
throw () [inline, virtual]

### 6.378.2 Member Function Documentation

6.378.2.1 virtual GeneralSecurityException\* decaf::security::GeneralSecurityException::clone ( ) const  
[inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

A deep copy of this exception.

Reimplemented from **decaf::lang::Exception** (p. 1889).

Reimplemented in **decaf::security::cert::CertificateEncodingException** (p. 1118), **decaf::security::cert::CertificateException** (p. 1120), **decaf::security::cert::CertificateExpiredException** (p. 1122), **decaf::security::cert::CertificateNotYetValidException** (p. 1124), **decaf::security::cert::CertificateParsingException** (p. 1126), **decaf::security::InvalidKeyException** (p. 2203), **decaf::security::KeyException** (p. 2368), **decaf::security::KeyManagementException** (p. 2371), **decaf::security::NoSuchAlgorithmException** (p. 2910), **decaf::security::NoSuchProviderException** (p. 2915), and **decaf::security::SignatureException** (p. 3604).

The documentation for this class was generated from the following file:

- src/main/decaf/security/GeneralSecurityException.h

## 6.379 decaf::internal::util::GenericResource< T > Class Template Reference

A Generic **Resource** (p. 3374) wraps some type and will delete it when the **Resource** (p. 3374) itself is deleted.

```
#include <src/main/decaf/internal/util/GenericResource.h>
```

Inheritance diagram for decaf::internal::util::GenericResource< T >:

### Public Member Functions

- **GenericResource** (T \*value)
- virtual ~**GenericResource** ()
- T \* **getManaged** () const
- void **setManaged** (T \*value)

### 6.379.1 Detailed Description

```
template<typename T> class decaf::internal::util::GenericResource< T >
```

A Generic **Resource** (p. 3374) wraps some type and will delete it when the **Resource** (p. 3374) itself is deleted.

Since

1.0

### 6.379.2 Constructor & Destructor Documentation

6.379.2.1 `template<typename T> decaf::internal::util::GenericResource< T >::GenericResource ( T * value ) [inline, explicit]`

6.379.2.2 `template<typename T> virtual decaf::internal::util::GenericResource< T >::~~GenericResource ( ) [inline, virtual]`

### 6.379.3 Member Function Documentation

6.379.3.1 `template<typename T> T* decaf::internal::util::GenericResource< T >::getManaged ( ) const [inline]`

6.379.3.2 `template<typename T> void decaf::internal::util::GenericResource< T >::setManaged ( T * value ) [inline]`

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/GenericResource.h`

## 6.380 gz\_header\_s Struct Reference

```
#include <src/main/decaf/internal/util/zip/zlib.h>
```

### Data Fields

- `int text`
- `uLong time`
- `int xflags`
- `int os`
- `Bytef * extra`
- `uInt extra_len`
- `uInt extra_max`
- `Bytef * name`
- `uInt name_max`



- **Bytef \* comment**
- **uInt comm\_max**
- **int hcrc**
- **int done**

### 6.380.1 Field Documentation

6.380.1.1 **uInt gz\_header\_s::comm\_max**

6.380.1.2 **Bytef\* gz\_header\_s::comment**

6.380.1.3 **int gz\_header\_s::done**

6.380.1.4 **Bytef\* gz\_header\_s::extra**

6.380.1.5 **uInt gz\_header\_s::extra\_len**

6.380.1.6 **uInt gz\_header\_s::extra\_max**

6.380.1.7 **int gz\_header\_s::hcrc**

6.380.1.8 **Bytef\* gz\_header\_s::name**

6.380.1.9 **uInt gz\_header\_s::name\_max**

6.380.1.10 **int gz\_header\_s::os**

6.380.1.11 **int gz\_header\_s::text**

6.380.1.12 **uLong gz\_header\_s::time**

6.380.1.13 **int gz\_header\_s::xflags**

The documentation for this struct was generated from the following file:

- `src/main/decaf/internal/util/zip/zlib.h`

## 6.381 gz\_state Struct Reference

```
#include <src/main/decaf/internal/util/zip/gzguts.h>
```

### Data Fields

- **int mode**
- **int fd**

- char \* **path**
- z\_off64\_t **pos**
- unsigned **size**
- unsigned **want**
- unsigned char \* **in**
- unsigned char \* **out**
- unsigned char \* **next**
- unsigned **have**
- int **eof**
- z\_off64\_t **start**
- z\_off64\_t **raw**
- int **how**
- int **direct**
- int **level**
- int **strategy**
- z\_off64\_t **skip**
- int **seek**
- int **err**
- char \* **msg**
- z\_stream **strm**

### 6.381.1 Field Documentation

- 6.381.1.1 `int gz_state::direct`
- 6.381.1.2 `int gz_state::eof`
- 6.381.1.3 `int gz_state::err`
- 6.381.1.4 `int gz_state::fd`
- 6.381.1.5 `unsigned gz_state::have`
- 6.381.1.6 `int gz_state::how`
- 6.381.1.7 `unsigned char* gz_state::in`
- 6.381.1.8 `int gz_state::level`
- 6.381.1.9 `int gz_state::mode`
- 6.381.1.10 `char* gz_state::msg`
- 6.381.1.11 `unsigned char* gz_state::next`
- 6.381.1.12 `unsigned char* gz_state::out`
- 6.381.1.13 `char* gz_state::path`
- 6.381.1.14 `z_off64_t gz_state::pos`
- 6.381.1.15 `z_off64_t gz_state::raw`
- 6.381.1.16 `int gz_state::seek`
- 6.381.1.17 `unsigned gz_state::size`
- 6.381.1.18 `z_off64_t gz_state::skip`
- 6.381.1.19 `z_off64_t gz_state::start`
- 6.381.1.20 `int gz_state::strategy`
- 6.381.1.21 `z_stream gz_state::strm`
- 6.381.1.22 `unsigned gz_state::want`

The documentation for this struct was generated from the following file:

- `src/main/decaf/internal/util/zip/gzguts.h`

## 6.382 decaf::util::logging::Handler Class Reference

A **Handler** (p. 2042) object takes log messages from a **Logger** (p. 2461) and exports them.

```
#include <src/main/decaf/util/logging/Handler.h>
```

Inheritance diagram for decaf::util::logging::Handler:

### Public Member Functions

- **Handler** ()
- virtual  $\sim$ **Handler** ()
- virtual void **flush** ()=0  
*Flush the Handler's output, clears any buffers.*
- virtual void **publish** (const **LogRecord** &record)=0  
*Publish the Log Record to this **Handler** (p. 2042).*
- virtual bool **isLoggable** (const **LogRecord** &record) const  
*Check if this **Handler** (p. 2042) would actually log a given **LogRecord** (p. 2487).*
- virtual void **setFilter** (**Filter** \*filter)  
*Sets the **Filter** (p. 1949) that this **Handler** (p. 2042) uses to filter Log Records.*
- virtual **Filter** \* **getFilter** ()  
*Gets the **Filter** (p. 1949) that this **Handler** (p. 2042) uses to filter Log Records.*
- virtual void **setLevel** (const **Level** &value)  
***Set** (p. 3538) the log level specifying which message levels will be logged by this **Handler** (p. 2042).*
- virtual **Level** **getLevel** ()  
*Get the log level specifying which message levels will be logged by this **Handler** (p. 2042).*
- virtual void **setFormatter** (**Formatter** \*formatter)  
*Sets the **Formatter** (p. 2027) used by this **Handler** (p. 2042).*
- virtual **Formatter** \* **getFormatter** ()  
*Gets the **Formatter** (p. 2027) used by this **Handler** (p. 2042).*
- virtual void **setErrorManager** (**ErrorManager** \*errorManager)  
*Sets the **Formatter** (p. 2027) used by this **Handler** (p. 2042).*

- virtual **ErrorManager** \* **getErrorHandler** ()

*Gets the **ErrorManager** (p. 1884) used by this **Handler** (p. 2042).*

### Protected Member Functions

- void **reportError** (const std::string &message, **decaf::lang::Exception** \*ex, int code)

*Protected convenience method to report an error to this **Handler**'s **ErrorManager** (p. 1884).*

### 6.382.1 Detailed Description

A **Handler** (p. 2042) object takes log messages from a **Logger** (p. 2461) and exports them. It might for example, write them to a console or write them to a file, or send them to a network logging service, or forward them to an OS log, or whatever.

A **Handler** (p. 2042) can be disabled by doing a **setLevel(Level.OFF** (p. 2408)) and can be re-enabled by doing a **setLevel** with an appropriate level.

**Handler** (p. 2042) classes typically use **LogManager** (p. 2480) properties to set default values for the **Handler**'s **Filter** (p. 1949), **Formatter** (p. 2027), and **Level** (p. 2403). See the specific documentation for each concrete **Handler** (p. 2042) class.

### 6.382.2 Constructor & Destructor Documentation

6.382.2.1 **decaf::util::logging::Handler::Handler** ( )

6.382.2.2 virtual **decaf::util::logging::Handler::~~Handler** ( ) [virtual]

### 6.382.3 Member Function Documentation

6.382.3.1 virtual void **decaf::util::logging::Handler::flush** ( ) [pure virtual]

Flush the **Handler**'s output, clears any buffers.

Implemented in **decaf::util::logging::StreamHandler** (p. 3759).

6.382.3.2 virtual **ErrorManager**\* **decaf::util::logging::Handler::getErrorHandler** ( )  
[inline, virtual]

Gets the **ErrorManager** (p. 1884) used by this **Handler** (p. 2042).

### Returns

**ErrorManager** (p. 1884) derived pointer or NULL.

**6.382.3.3** `virtual Filter* decaf::util::logging::Handler::getFilter ( ) [inline, virtual]`

Gets the **Filter** (p. 1949) that this **Handler** (p. 2042) uses to filter Log Records.

#### Returns

**Filter** (p. 1949) derived instance

**6.382.3.4** `virtual Formatter* decaf::util::logging::Handler::getFormatter ( ) [inline, virtual]`

Gets the **Formatter** (p. 2027) used by this **Handler** (p. 2042).

#### Returns

**Filter** (p. 1949) derived instance

**6.382.3.5** `virtual Level decaf::util::logging::Handler::getLevel ( ) [inline, virtual]`

Get the log level specifying which message levels will be logged by this **Handler** (p. 2042).

#### Returns

**Level** (p. 2403) enumeration value

**6.382.3.6** `virtual bool decaf::util::logging::Handler::isLoggable ( const LogRecord & record ) const [virtual]`

Check if this **Handler** (p. 2042) would actually log a given **LogRecord** (p. 2487).

This method checks if the **LogRecord** (p. 2487) has an appropriate **Level** (p. 2403) and whether it satisfies any **Filter** (p. 1949). It also may make other **Handler** (p. 2042) specific checks that might prevent a handler from logging the **LogRecord** (p. 2487).

#### Parameters

<i>record</i>	<b>LogRecord</b> (p. 2487) to check
---------------	-------------------------------------

Reimplemented in **decaf::util::logging::StreamHandler** (p. 3759).

**6.382.3.7** virtual void decaf::util::logging::Handler::publish ( const LogRecord & *record* )  
[pure virtual]

Publish the Log Record to this **Handler** (p. 2042).

#### Parameters

<i>record</i>	The Log Record to Publish
---------------	---------------------------

Implemented in **decaf::util::logging::ConsoleHandler** (p. 1441), and **decaf::util::logging::StreamHandler** (p. 3759).

**6.382.3.8** void decaf::util::logging::Handler::reportError ( const std::string & *message*,  
decaf::lang::Exception \* *ex*, int *code* ) [protected]

Protected convenience method to report an error to this Handler's **ErrorManager** (p. 1884).

#### Parameters

<i>message</i>	- a descriptive string (may be empty)
<i>ex</i>	- an exception (may be NULL)
<i>code</i>	- an error code defined in <b>ErrorManager</b> (p. 1884)

**6.382.3.9** virtual void decaf::util::logging::Handler::setErrorHandler ( **ErrorManager** \*  
*errorManager* ) [virtual]

Sets the **Formatter** (p. 2027) used by this **Handler** (p. 2042).

The **ErrorManager**'s "error" method will be invoked if any errors occur while using this **Handler** (p. 2042).

#### Parameters

<i>errorManager</i>	<b>ErrorManager</b> (p. 1884) derived instance
---------------------	--

**6.382.3.10** virtual void decaf::util::logging::Handler::setFilter ( **Filter** \* *filter* ) [inline,  
virtual]

Sets the **Filter** (p. 1949) that this **Handler** (p. 2042) uses to filter Log Records.

For each call of publish the **Handler** (p. 2042) will call this **Filter** (p. 1949) (if it is non-null) to check if the **LogRecord** (p. 2487) should be published or discarded.

#### Parameters

<i>filter</i>	<b>Filter</b> (p. 1949) derived instance
---------------	--

**6.382.3.11** `virtual void decaf::util::logging::Handler::setFormatter ( Formatter * formatter )`  
`[virtual]`

Sets the **Formatter** (p. 2027) used by this **Handler** (p. 2042).

Some Handlers may not use Formatters, in which case the **Formatter** (p. 2027) will be remembered, but not used.

#### Parameters

<i>formatter</i>	<b>Filter</b> (p. 1949) derived instance
------------------	--

**6.382.3.12** `virtual void decaf::util::logging::Handler::setLevel ( const Level & value )`  
`[inline, virtual]`

**Set** (p. 3538) the log level specifying which message levels will be logged by this **Handler** (p. 2042).

The intention is to allow developers to turn on voluminous logging, but to limit the messages that are sent to certain Handlers.

#### Parameters

<i>value</i>	<b>Level</b> (p. 2403) enumeration value
--------------	--

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/Handler.h`

## 6.383 decaf::internal::util::HexStringParser Class Reference

```
#include <src/main/decaf/internal/util/HexStringParser.h>
```

### Public Member Functions

- **HexStringParser** (int exponentWidth, int mantissaWidth)  
*Create a new HexParser.*
- virtual **~HexStringParser** ()
- long long **parse** (const std::string &hexString)  
*Parses a hex string using the specs given in the constructor and returns a long long with the bits of the parsed string, the caller can then convert those to a float or double as needed.*

### Static Public Member Functions

- static double **parseDouble** (const std::string &hexString)



- static float **parseFloat** (const std::string &hexString)

### 6.383.1 Constructor & Destructor Documentation

#### 6.383.1.1 decaf::internal::util::HexStringParser::HexStringParser ( int *exponentWidth*, int *mantissaWidth* )

Create a new HexParser.

##### Parameters

<i>exponentWidth</i>	- Width of the exponent for the type to parse
<i>mantissaWidth</i>	- Width of the mantissa for the type to parse

#### 6.383.1.2 virtual decaf::internal::util::HexStringParser::~~HexStringParser ( ) [inline, virtual]

### 6.383.2 Member Function Documentation

#### 6.383.2.1 long long decaf::internal::util::HexStringParser::parse ( const std::string & *hexString* )

Parses a hex string using the specs given in the constructor and returns a long long with the bits of the parsed string, the caller can then convert those to a float or double as needed.

##### Parameters

<i>hexString</i>	- string to parse
------------------	-------------------

##### Returns

the bits parsed from the string

#### 6.383.2.2 static double decaf::internal::util::HexStringParser::parseDouble ( const std::string & *hexString* ) [static]

#### 6.383.2.3 static float decaf::internal::util::HexStringParser::parseFloat ( const std::string & *hexString* ) [static]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**HexStringParser.h**

## 6.384 activemq::wireformat::openwire::utils::HexTable Class Reference

The **HexTable** (p. 2048) class maps hexadecimal strings to the value of an index into the table, i.e.

```
#include <src/main/activemq/wireformat/openwire/utils/HexTable.h>
```

### Public Member Functions

- **HexTable** ()
- virtual **~HexTable** ()
- virtual const std::string & **operator[]** (std::size\_t index) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Index operator for this Table, will throw an exception if the index requested is out of bounds for this table.*
- virtual const std::string & **operator[]** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual std::size\_t **size** () const  
*Returns the max size of this Table.*

### 6.384.1 Detailed Description

The **HexTable** (p. 2048) class maps hexadecimal strings to the value of an index into the table, i.e. the class will return "FF" for the index 255 in the table.

### 6.384.2 Constructor & Destructor Documentation

6.384.2.1 **activemq::wireformat::openwire::utils::HexTable::HexTable** ( )

6.384.2.2 **virtual activemq::wireformat::openwire::utils::HexTable::~~HexTable** ( )  
[inline, virtual]

### 6.384.3 Member Function Documentation

6.384.3.1 **virtual const std::string& activemq::wireformat::openwire::utils::HexTable::operator[]** ( **std::size\_t index** ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Index operator for this Table, will throw an exception if the index requested is out of bounds for this table.

### Parameters

<i>index</i>	The index of the value in the table to fetch.
--------------	---

**Returns**

string containing the hex value if the index

**Exceptions**

<i>IndexOutOfBoundsException</i>	
----------------------------------	--

**6.384.3.2** virtual const std::string& activemq::wireformat::openwire::utils::HexTable::operator[]  
( std::size\_t *index* ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

**6.384.3.3** virtual std::size\_t activemq::wireformat::openwire::utils::HexTable::size ( ) const  
[inline, virtual]

Returns the max size of this Table.

**Returns**

an integer size value

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/utils/**HexTable.h**

**6.385 decaf::net::HttpRetryException Class Reference**

```
#include <src/main/decaf/net/HttpRetryException.h>
```

Inheritance diagram for decaf::net::HttpRetryException:

**Public Member Functions**

- **HttpRetryException** () throw ()  
*Default Constructor.*
- **HttpRetryException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **HttpRetryException** (const **HttpRetryException** &ex) throw ()  
*Copy Constructor.*

- **HttpException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **HttpException** (const std::exception \*cause) throw ()  
*Constructor.*
- **HttpException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **HttpException \* clone** () const  
*Clones this exception.*
- virtual ~**HttpException** () throw ()

### 6.385.1 Constructor & Destructor Documentation

6.385.1.1 **decaf::net::HttpException::HttpException ( ) throw ()** [inline]

Default Constructor.

6.385.1.2 **decaf::net::HttpException::HttpException ( const Exception & ex ) throw ()** [inline]

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.385.1.3 **decaf::net::HttpException::HttpException ( const HttpException & ex ) throw ()** [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.385.1.4 **decaf::net::HttpException::HttpException ( const char \* file, const int lineNumber, const std::exception \* cause, const char \* msg, ... ) throw ()** [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.385.1.5** decaf::net::HttpRetryException::HttpRetryException ( const std::exception \* *cause* )  
throw () [inline]

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.385.1.6** decaf::net::HttpRetryException::HttpRetryException ( const char \* *file*, const int *lineNumber*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.385.1.7** virtual decaf::net::HttpRetryException::~~HttpRetryException ( ) throw ()  
[inline, virtual]

## 6.385.2 Member Function Documentation

**6.385.2.1** virtual HttpRetryException\* decaf::net::HttpRetryException::clone ( ) const  
[inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

a new `Exception` instance that is a copy of this `Exception` object.

Reimplemented from `decaf::io::IOException` (p. 2212).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/HttpRetryException.h`

## 6.386 `activemq::util::IdGenerator` Class Reference

```
#include <src/main/activemq/util/IdGenerator.h>
```

### Data Structures

- class `StaticData`

### Public Member Functions

- `IdGenerator ()`
- `IdGenerator (const std::string &prefix)`
- `virtual ~IdGenerator ()`
- `std::string generateId () const`

### Static Public Member Functions

- `static std::string getHostname ()`  
*Since the initialization of this object results in the retrieval of the machine's host name we can quickly return it here.*
- `static std::string getSeedFromId (const std::string &id)`  
*Gets the seed value from a Generated Id, the count portion is removed.*
- `static long long getSequenceFromId (const std::string &id)`  
*Gets the count value from a Generated Id, the seed portion is removed.*
- `static int compare (const std::string &id1, const std::string &id2)`  
*Compares two generated id values.*

## 6.386.1 Constructor & Destructor Documentation

6.386.1.1 `activemq::util::IdGenerator::IdGenerator ( )`

6.386.1.2 `activemq::util::IdGenerator::IdGenerator ( const std::string & prefix )`

6.386.1.3 `virtual activemq::util::IdGenerator::~~IdGenerator ( )` `[virtual]`

## 6.386.2 Member Function Documentation

6.386.2.1 `static int activemq::util::IdGenerator::compare ( const std::string & id1, const std::string & id2 )` `[static]`

Compares two generated id values.

### Parameters

<i>id1</i>	The first id to compare, or left hand side.
<i>id2</i>	The second id to compare, or right hand side.

### Returns

zero if ids are equal or positive if *id1* > *id2*...

6.386.2.2 `std::string activemq::util::IdGenerator::generateId ( )` `const`

### Returns

a newly generated unique id.

6.386.2.3 `static std::string activemq::util::IdGenerator::getHostname ( )` `[static]`

Since the initialization of this object results in the retrieval of the machine's host name we can quickly return it here.

### Returns

the previously retrieved host name.

6.386.2.4 `static std::string activemq::util::IdGenerator::getSeedFromId ( const std::string & id )` `[static]`

Gets the seed value from a Generated Id, the count portion is removed.

### Returns

the seed portion of the Id, minus the count value.

**6.386.2.5** `static long long activemq::util::IdGenerator::getSequenceFromId ( const std::string & id ) [static]`

Gets the count value from a Generated Id, the seed portion is removed.

#### Returns

the sequence count portion of the id, minus the seed value.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/IdGenerator.h`

## 6.387 decaf::lang::exceptions::IllegalArgumentException Class Reference

```
#include <src/main/decaf/lang/exceptions/IllegalArgumentException.h>
```

Inheritance diagram for `decaf::lang::exceptions::IllegalArgumentException`:

### Public Member Functions

- **IllegalArgumentException** () throw ()  
*Default Constructor.*
- **IllegalArgumentException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1886).*
- **IllegalArgumentException** (const **IllegalArgumentException** &ex) throw ()  
*Copy Constructor.*
- **IllegalArgumentException** (const std::exception \***cause**) throw ()  
*Constructor.*
- **IllegalArgumentException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IllegalArgumentException** (const char \*file, const int lineNumber, const std::exception \***cause**, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **IllegalArgumentException** \* **clone** () const  
*Clones this exception.*
- virtual ~**IllegalArgumentException** () throw ()



### 6.387.1 Constructor & Destructor Documentation

6.387.1.1 `decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException ( )  
throw () [inline]`

Default Constructor.

6.387.1.2 `decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException ( const  
Exception & ex ) throw () [inline]`

Conversion Constructor from some other **Exception** (p. 1886).

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

6.387.1.3 `decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException ( const  
IllegalArgumentException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

6.387.1.4 `decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException ( const  
std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	<b>Pointer</b> (p. 3032) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.387.1.5 `decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException ( const  
char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report

...	list of primitives that are formatted into the message
-----	--

**6.387.1.6** `decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.387.1.7** `virtual decaf::lang::exceptions::IllegalArgumentException::~~IllegalArgumentException ( ) throw () [inline, virtual]`

## 6.387.2 Member Function Documentation

**6.387.2.1** `virtual IllegalArgumentException* decaf::lang::exceptions::IllegalArgumentException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

an new **Exception** (p. 1886) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1889).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IllegalArgumentException.h`

## 6.388 decaf::lang::exceptions::IllegalMonitorStateException Class Reference

```
#include <src/main/decaf/lang/exceptions/IllegalMonitorStateException.h>
```

## 6.388 decaf::lang::exceptions::IllegalMonitorStateException Class Reference 2059

Inheritance diagram for decaf::lang::exceptions::IllegalMonitorStateException:

### Public Member Functions

- **IllegalMonitorStateException** () throw ()  
*Default Constructor.*
- **IllegalMonitorStateException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1886).*
- **IllegalMonitorStateException** (const **IllegalMonitorStateException** &ex) throw ()  
*Copy Constructor.*
- **IllegalMonitorStateException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IllegalMonitorStateException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IllegalMonitorStateException** (const std::exception \*cause) throw ()  
*Constructor.*
- virtual **IllegalMonitorStateException** \* clone () const  
*Clones this exception.*
- virtual ~**IllegalMonitorStateException** () throw ()

### 6.388.1 Constructor & Destructor Documentation

6.388.1.1 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException ( ) throw () [inline]

Default Constructor.

6.388.1.2 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException ( const **Exception** & ex ) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1886).

### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

**6.388.1.3** `decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException ( const IllegalMonitorStateException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

**6.388.1.4** `decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.388.1.5** `decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.388.1.6** `decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	<b>Pointer</b> (p. 3032) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.388.1.7 virtual decaf::lang::exceptions::IllegalMonitorStateException::~~IllegalMonitorStateException ( ) throw () [inline, virtual]

## 6.388.2 Member Function Documentation

6.388.2.1 virtual IllegalStateException\* decaf::lang::exceptions::IllegalMonitorStateException::clone ( ) const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

an new **Exception** (p. 1886) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1889).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**IllegalMonitorStateException.h**

## 6.389 cms::IllegalStateException Class Reference

This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.

```
#include <src/main/cms/IllegalStateException.h>
```

Inheritance diagram for cms::IllegalStateException:

### Public Member Functions

- **IllegalStateException** () throw ()
- **IllegalStateException** (const **IllegalStateException** &ex) throw ()
- **IllegalStateException** (const std::string &message, const std::exception \*cause) throw ()
- **IllegalStateException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**IllegalStateException** () throw ()

### 6.389.1 Detailed Description

This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation. For example,

this exception must be thrown if **Session.commit** (p. 3465) is called on a non-transacted session.

Since

1.3

## 6.389.2 Constructor & Destructor Documentation

6.389.2.1 `cms::IllegalStateException::IllegalStateException ( ) throw ()`

6.389.2.2 `cms::IllegalStateException::IllegalStateException ( const IllegalStateException & ex ) throw ()`

6.389.2.3 `cms::IllegalStateException::IllegalStateException ( const std::string & message, const std::exception * cause ) throw ()`

6.389.2.4 `cms::IllegalStateException::IllegalStateException ( const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace ) throw ()`

6.389.2.5 `virtual cms::IllegalStateException::~~IllegalStateException ( ) throw ()`  
[virtual]

The documentation for this class was generated from the following file:

- `src/main/cms/IllegalStateException.h`

## 6.390 decaf::lang::exceptions::IllegalStateException Class Reference

```
#include <src/main/decaf/lang/exceptions/IllegalStateException.h>
```

Inheritance diagram for `decaf::lang::exceptions::IllegalStateException`:

### Public Member Functions

- **IllegalStateException** ( ) throw ()  
*Default Constructor.*
- **IllegalStateException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1886).*
- **IllegalStateException** (const **IllegalStateException** &ex) throw ()  
*Copy Constructor.*

- **IllegalStateException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IllegalStateException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IllegalStateException** (const std::exception \*cause) throw ()  
*Constructor.*
- virtual **IllegalStateException** \* clone () const  
*Clones this exception.*
- virtual ~**IllegalStateException** () throw ()

### 6.390.1 Constructor & Destructor Documentation

6.390.1.1 decaf::lang::exceptions::IllegalStateException::IllegalStateException ( ) throw ()  
[inline]

Default Constructor.

6.390.1.2 decaf::lang::exceptions::IllegalStateException::IllegalStateException ( const Exception & ex ) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1886).

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

6.390.1.3 decaf::lang::exceptions::IllegalStateException::IllegalStateException ( const IllegalStateException & ex ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

6.390.1.4 decaf::lang::exceptions::IllegalStateException::IllegalStateException ( const char \* file, const int lineNumber, const char \* msg, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.390.1.5** `decaf::lang::exceptions::IllegalStateException::IllegalStateException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.390.1.6** `decaf::lang::exceptions::IllegalStateException::IllegalStateException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	<b>Pointer</b> (p.3032) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.390.1.7** `virtual decaf::lang::exceptions::IllegalStateException::~~IllegalStateException ( ) throw () [inline, virtual]`

## 6.390.2 Member Function Documentation

**6.390.2.1** `virtual IllegalStateException* decaf::lang::exceptions::IllegalStateException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.



## 6.391 decaf::lang::exceptions::IllegalThreadStateException Class Reference 2065

### Returns

an new **Exception** (p. 1886) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1889).

Reimplemented in **decaf::nio::InvalidMarkException** (p. 2206).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**IllegalStateException.h**

## 6.391 decaf::lang::exceptions::IllegalThreadStateException Class Reference

```
#include <src/main/decaf/lang/exceptions/IllegalThreadStateException.h>
```

Inheritance diagram for decaf::lang::exceptions::IllegalThreadStateException:

### Public Member Functions

- **IllegalThreadStateException** () throw ()  
*Default Constructor.*
- **IllegalThreadStateException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1886).*
- **IllegalThreadStateException** (const **IllegalThreadStateException** &ex) throw ()  
*Copy Constructor.*
- **IllegalThreadStateException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IllegalThreadStateException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IllegalThreadStateException** (const std::exception \*cause) throw ()  
*Constructor.*
- virtual **IllegalThreadStateException** \* **clone** () const  
*Clones this exception.*
- virtual ~**IllegalThreadStateException** () throw ()

### 6.391.1 Constructor & Destructor Documentation

6.391.1.1 `decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException ( )  
throw () [inline]`

Default Constructor.

6.391.1.2 `decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException ( const Exception & ex ) throw () [inline]`

Conversion Constructor from some other **Exception** (p. 1886).

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

6.391.1.3 `decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException ( const IllegalThreadStateException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

6.391.1.4 `decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.391.1.5 `decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.391.1.6 `decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const std::exception * cause ) throw () [inline]`

Constructor.

**Parameters**

<i>cause</i>	<b>Pointer</b> (p. 3032) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.391.1.7 `virtual decaf::lang::exceptions::IllegalThreadStateException::~~IllegalThreadStateException ( ) throw () [inline, virtual]`

**6.391.2 Member Function Documentation**

6.391.2.1 `virtual IllegalThreadStateException* decaf::lang::exceptions::IllegalThreadStateException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns**

an new **Exception** (p. 1886) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1889).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IllegalThreadStateException.h`

**6.392 activemq::transport::inactivity::InactivityMonitor Class Reference**

```
#include <src/main/activemq/transport/inactivity/InactivityMonitor.h>
```

Inheritance diagram for activemq::transport::inactivity::InactivityMonitor:

## Public Member Functions

- **InactivityMonitor** (const **Pointer**< **Transport** > &next, const **Pointer**< **wireformat::WireFormat** > &wireFormat)
- **InactivityMonitor** (const **Pointer**< **Transport** > &next, const **decaf::util::Properties** &properties, const **Pointer**< **wireformat::WireFormat** > &wireFormat)
- virtual ~**InactivityMonitor** ()
- virtual void **close** () throw ( decaf::io::IOException )  
*Stops the polling thread and closes the streams.*
- virtual void **onException** (const **decaf::lang::Exception** &ex)  
*Event handler for an exception from a command transport.*
- virtual void **onCommand** (const **Pointer**< **Command** > &command)  
*Event handler for the receipt of a command.*
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends a one-way command.*
- bool **isKeepAliveResponseRequired** () const
- void **setKeepAliveResponseRequired** (bool value)
- long long **getReadCheckTime** () const
- void **setReadCheckTime** (long long value)
- long long **getWriteCheckTime** () const
- void **setWriteCheckTime** (long long value)
- long long **getInitialDelayTime** () const
- void **setInitialDelayTime** (long long value) const

## Friends

- class **ReadChecker**
- class **AsyncSignalReadErrorTask**
- class **WriteChecker**
- class **AsyncWriteTask**

### 6.392.1 Constructor & Destructor Documentation

6.392.1.1 `activemq::transport::inactivity::InactivityMonitor ( const Pointer< Transport > & next, const Pointer< wireformat::WireFormat > & wireFormat )`

6.392.1.2 `activemq::transport::inactivity::InactivityMonitor ( const Pointer< Transport > & next, const decaf::util::Properties & properties, const Pointer< wireformat::WireFormat > & wireFormat )`

6.392.1.3 `virtual activemq::transport::inactivity::InactivityMonitor::~InactivityMonitor ( )`  
[virtual]

### 6.392.2 Member Function Documentation

6.392.2.1 `virtual void activemq::transport::inactivity::InactivityMonitor::close ( ) throw ( decaf::io::IOException )` [virtual]

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

#### Exceptions

<i>IOException</i>	if an error occurs while closing the <b>Transport</b> (p. 3996).
--------------------	--

Reimplemented from `activemq::transport::TransportFilter` (p. 4007).

6.392.2.2 `long long activemq::transport::inactivity::InactivityMonitor::getInitialDelayTime ( )`  
const

6.392.2.3 `long long activemq::transport::inactivity::InactivityMonitor::getReadCheckTime ( )`  
const

6.392.2.4 `long long activemq::transport::inactivity::InactivityMonitor::getWriteCheckTime ( )`  
const

6.392.2.5 `bool activemq::transport::inactivity::InactivityMonitor::isKeepAliveResponseRequired ( )` const

6.392.2.6 `virtual void activemq::transport::inactivity::InactivityMonitor::onCommand ( const Pointer< Command > & command )` [virtual]

Event handler for the receipt of a command.

#### Parameters

<i>command</i>	- the received command object.
----------------	--------------------------------

Reimplemented from **activemq::transport::TransportFilter** (p. 4009).

**6.392.2.7** `virtual void activemq::transport::inactivity::InactivityMonitor::oneway ( const  
Pointer< Command > & command ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

#### Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

#### Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 4010).

**6.392.2.8** `virtual void activemq::transport::inactivity::InactivityMonitor::onException ( const  
decaf::lang::Exception & ex ) [virtual]`

Event handler for an exception from a command transport.

#### Parameters

<i>ex</i>	The exception to handle.
-----------	--------------------------

Reimplemented from **activemq::transport::TransportFilter** (p. 4010).

## 6.393 decaf::lang::exceptions::IndexOutOfBoundsException Class Reference 2071

- 6.392.2.9 void activemq::transport::inactivity::InactivityMonitor::setInitialDelayTime ( long long *value* ) const
- 6.392.2.10 void activemq::transport::inactivity::InactivityMonitor::setKeepAliveResponseRequired ( bool *value* )
- 6.392.2.11 void activemq::transport::inactivity::InactivityMonitor::setReadCheckTime ( long long *value* )
- 6.392.2.12 void activemq::transport::inactivity::InactivityMonitor::setWriteCheckTime ( long long *value* )

### 6.392.3 Friends And Related Function Documentation

- 6.392.3.1 friend class AsyncSignalReadErrorTask [friend]
- 6.392.3.2 friend class AsyncWriteTask [friend]
- 6.392.3.3 friend class ReadChecker [friend]
- 6.392.3.4 friend class WriteChecker [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/transport/inactivity/**InactivityMonitor.h**

## 6.393 decaf::lang::exceptions::IndexOutOfBoundsException Class Reference

```
#include <src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Inheritance diagram for decaf::lang::exceptions::IndexOutOfBoundsException:

### Public Member Functions

- **IndexOutOfBoundsException** () throw ()  
*Default Constructor.*
- **IndexOutOfBoundsException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1886).*
- **IndexOutOfBoundsException** (const **IndexOutOfBoundsException** &ex) throw ()  
*Copy Constructor.*

- **IndexOutOfBoundsException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IndexOutOfBoundsException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IndexOutOfBoundsException** (const std::exception \*cause) throw ()  
*Constructor.*
- virtual **IndexOutOfBoundsException** \* clone () const  
*Clones this exception.*
- virtual ~**IndexOutOfBoundsException** () throw ()

### 6.393.1 Constructor & Destructor Documentation

6.393.1.1 `decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException ( ) throw () [inline]`

Default Constructor.

6.393.1.2 `decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException ( const Exception & ex ) throw () [inline]`

Conversion Constructor from some other **Exception** (p. 1886).

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

6.393.1.3 `decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException ( const IndexOutOfBoundsException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--



## 6.393 decaf::lang::exceptions::IndexOutOfBoundsException Class Reference 2073

6.393.1.4 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException ( const char \* *file*, const int *lineNumber*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.393.1.5 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException ( const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.393.1.6 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException ( const std::exception \* *cause* ) throw () [inline]

Constructor.

### Parameters

<i>cause</i>	<b>Pointer</b> (p. 3032) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.393.1.7 virtual `decaf::lang::exceptions::IndexOutOfBoundsException::~IndexOutOfBoundsException ( ) throw ( )` `[inline, virtual]`

### 6.393.2 Member Function Documentation

6.393.2.1 virtual `IndexOutOfBoundsException* decaf::lang::exceptions::IndexOutOfBoundsException::clone ( ) const` `[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

an new **Exception** (p. 1886) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1889).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h`

## 6.394 decaf::net::Inet4Address Class Reference

```
#include <src/main/decaf/net/Inet4Address.h>
```

Inheritance diagram for `decaf::net::Inet4Address`:

#### Public Member Functions

- virtual `~Inet4Address ( )`
- virtual bool `isAnyLocalAddress ( ) const`  
*Check if this **InetAddress** (p. 2077) is a valid wildcard address.*
- virtual bool `isLoopbackAddress ( ) const`  
*Check if this **InetAddress** (p. 2077) is a valid loopback address.*
- virtual bool `isMulticastAddress ( ) const`  
*Check if this **InetAddress** (p. 2077) is a valid Multicast address.*
- virtual bool `isLinkLocalAddress ( ) const`  
*Check if this **InetAddress** (p. 2077) is a valid link local address.*
- virtual bool `isSiteLocalAddress ( ) const`

Check if this **InetAddress** (p. 2077) is a valid site local address.

- virtual bool **isMCGlobal** () const

Check if this **InetAddress** (p. 2077) is Multicast and has Global scope.

- virtual bool **isMCNodeLocal** () const

Check if this **InetAddress** (p. 2077) is Multicast and has Node Local scope.

- virtual bool **isMCLinkLocal** () const

Check if this **InetAddress** (p. 2077) is Multicast and has Link Local scope.

- virtual bool **isMCSiteLocal** () const

Check if this **InetAddress** (p. 2077) is Multicast and has Site Local scope.

- virtual bool **isMCOrgLocal** () const

Check if this **InetAddress** (p. 2077) is Multicast and has Organization scope.

## Protected Member Functions

- **Inet4Address** ()
- **Inet4Address** (const unsigned char \*ipAddress, int numBytes)
- **Inet4Address** (const std::string &hostname, const unsigned char \*ipAddress, int numBytes)

## Friends

- class **InetAddress**

## 6.394.1 Constructor & Destructor Documentation

6.394.1.1 decaf::net::Inet4Address::Inet4Address ( ) [protected]

6.394.1.2 decaf::net::Inet4Address::Inet4Address ( const unsigned char \* ipAddress, int numBytes ) [protected]

6.394.1.3 decaf::net::Inet4Address::Inet4Address ( const std::string & hostname, const unsigned char \* ipAddress, int numBytes ) [protected]

6.394.1.4 virtual decaf::net::Inet4Address::~~Inet4Address ( ) [virtual]

## 6.394.2 Member Function Documentation

6.394.2.1 virtual bool decaf::net::Inet4Address::isAnyLocalAddress ( ) const [virtual]

Check if this **InetAddress** (p. 2077) is a valid wildcard address.

**Returns**

true if the address is a wildcard address.

Reimplemented from **decaf::net::InetAddress** (p. 2082).

**6.394.2.2 virtual bool decaf::net::Inet4Address::isLinkLocalAddress ( ) const [virtual]**

Check if this **InetAddress** (p. 2077) is a valid link local address.

**Returns**

true if the address is a link local address.

Reimplemented from **decaf::net::InetAddress** (p. 2082).

**6.394.2.3 virtual bool decaf::net::Inet4Address::isLoopbackAddress ( ) const [virtual]**

Check if this **InetAddress** (p. 2077) is a valid loopback address.

**Returns**

true if the address is a loopback address.

Reimplemented from **decaf::net::InetAddress** (p. 2083).

**6.394.2.4 virtual bool decaf::net::Inet4Address::isMCGlobal ( ) const [virtual]**

Check if this **InetAddress** (p. 2077) is Multicast and has Global scope.

**Returns**

true if the address is Multicast and has Global scope.

Reimplemented from **decaf::net::InetAddress** (p. 2083).

**6.394.2.5 virtual bool decaf::net::Inet4Address::isMCLinkLocal ( ) const [virtual]**

Check if this **InetAddress** (p. 2077) is Multicast and has Link Local scope.

**Returns**

true if the address is Multicast and has Link Local scope.

Reimplemented from **decaf::net::InetAddress** (p. 2083).

**6.394.2.6** virtual bool decaf::net::Inet4Address::isMCNodeLocal ( ) const [virtual]

Check if this **InetAddress** (p. 2077) is Multicast and has Node Local scope.

#### Returns

true if the address is Multicast and has Node Local scope.

Reimplemented from **decaf::net::InetAddress** (p. 2083).

**6.394.2.7** virtual bool decaf::net::Inet4Address::isMCOrgLocal ( ) const [virtual]

Check if this **InetAddress** (p. 2077) is Multicast and has Organization scope.

#### Returns

true if the address is Multicast and has Organization scope.

Reimplemented from **decaf::net::InetAddress** (p. 2083).

**6.394.2.8** virtual bool decaf::net::Inet4Address::isMCSiteLocal ( ) const [virtual]

Check if this **InetAddress** (p. 2077) is Multicast and has Site Local scope.

#### Returns

true if the address is Multicast and has Site Local scope.

Reimplemented from **decaf::net::InetAddress** (p. 2084).

**6.394.2.9** virtual bool decaf::net::Inet4Address::isMulticastAddress ( ) const [virtual]

Check if this **InetAddress** (p. 2077) is a valid Multicast address.

#### Returns

true if the address is a Multicast address.

Reimplemented from **decaf::net::InetAddress** (p. 2084).

**6.394.2.10** virtual bool decaf::net::Inet4Address::isSiteLocalAddress ( ) const [virtual]

Check if this **InetAddress** (p. 2077) is a valid site local address.

#### Returns

true if the address is a site local address.

Reimplemented from **decaf::net::InetAddress** (p. 2084).

### 6.394.3 Friends And Related Function Documentation

#### 6.394.3.1 friend class **InetAddress** [friend]

The documentation for this class was generated from the following file:

- `src/main/decaf/net/Inet4Address.h`

### 6.395 **decaf::net::Inet6Address** Class Reference

```
#include <src/main/decaf/net/Inet6Address.h>
```

Inheritance diagram for `decaf::net::Inet6Address`:

#### Public Member Functions

- `virtual ~Inet6Address ()`

#### Protected Member Functions

- `Inet6Address ()`
- `Inet6Address (const unsigned char *ipAddress, int numBytes)`
- `Inet6Address (const std::string &hostname, const unsigned char *ipAddress, int numBytes)`

#### Friends

- class **InetAddress**

### 6.395.1 Constructor & Destructor Documentation

6.395.1.1 `decaf::net::Inet6Address::Inet6Address ( )` [protected]

6.395.1.2 `decaf::net::Inet6Address::Inet6Address ( const unsigned char * ipAddress, int numBytes )` [protected]

6.395.1.3 `decaf::net::Inet6Address::Inet6Address ( const std::string & hostname, const unsigned char * ipAddress, int numBytes )` [protected]

6.395.1.4 `virtual decaf::net::Inet6Address::~~Inet6Address ( )` [virtual]

### 6.395.2 Friends And Related Function Documentation

6.395.2.1 `friend class InetAddress` [friend]

The documentation for this class was generated from the following file:

- `src/main/decaf/net/Inet6Address.h`

## 6.396 decaf::net::InetAddress Class Reference

Represents an IP address.

```
#include <src/main/decaf/net/InetAddress.h>
```

Inheritance diagram for `decaf::net::InetAddress`:

### Public Member Functions

- `virtual ~InetAddress ()`
- `virtual decaf::lang::ArrayPointer< unsigned char > getAddress () const`  
*Returns the Raw IP address in Network byte order.*
- `virtual std::string getHostAddress () const`  
*Returns a textual representation of the IP Address.*
- `virtual std::string getHostName () const`  
*Get the host name associated with this **InetAddress** (p. 2077) instance.*
- `virtual std::string toString () const`  
*Returns a string representation of the **InetAddress** (p. 2077) in the form 'hostname / ipaddress'.*
- `virtual bool isAnyLocalAddress () const`

*Check if this **InetAddress** (p. 2077) is a valid wildcard address.*

- virtual bool **isLoopbackAddress** () const

*Check if this **InetAddress** (p. 2077) is a valid loopback address.*

- virtual bool **isMulticastAddress** () const

*Check if this **InetAddress** (p. 2077) is a valid Multicast address.*

- virtual bool **isLinkLocalAddress** () const

*Check if this **InetAddress** (p. 2077) is a valid link local address.*

- virtual bool **isSiteLocalAddress** () const

*Check if this **InetAddress** (p. 2077) is a valid site local address.*

- virtual bool **isMCGlobal** () const

*Check if this **InetAddress** (p. 2077) is Multicast and has Global scope.*

- virtual bool **isMCNodeLocal** () const

*Check if this **InetAddress** (p. 2077) is Multicast and has Node Local scope.*

- virtual bool **isMCLinkLocal** () const

*Check if this **InetAddress** (p. 2077) is Multicast and has Link Local scope.*

- virtual bool **isMCSiteLocal** () const

*Check if this **InetAddress** (p. 2077) is Multicast and has Site Local scope.*

- virtual bool **isMCOrgLocal** () const

*Check if this **InetAddress** (p. 2077) is Multicast and has Organization scope.*

## Static Public Member Functions

- static **InetAddress** **getByAddress** (const unsigned char \*bytes, int numBytes)

*Given a raw IP Address in byte array form, create and return a new **InetAddress** (p. 2077) instance.*

- static **InetAddress** **getByAddress** (const std::string &hostname, const unsigned char \*bytes, int numBytes)

*Given a host name and IPAddress return a new **InetAddress** (p. 2077).*

- static **InetAddress** **getLocalHost** ()

*Gets an **InetAddress** (p. 2077) that is the local host address.*



### Protected Member Functions

- **InetAddress** ()
- **InetAddress** (const unsigned char \*ipAddress, int numBytes)
- **InetAddress** (const std::string &hostname, const unsigned char \*ipAddress, int numBytes)

### Static Protected Member Functions

- static unsigned int **bytesToInt** (const unsigned char \*bytes, int start)

*Converts the bytes in an address array to an int starting from the value start treating the start value as the high order byte.*

- static **InetAddress** **getAnyAddress** ()
- static **InetAddress** **getLoopbackAddress** ()

### Protected Attributes

- std::string **hostname**
- bool **reached**
- decaf::lang::ArrayPointer< unsigned char > **addressBytes**

### Static Protected Attributes

- static const unsigned char **loopbackBytes** [4]
- static const unsigned char **anyBytes** [4]

#### 6.396.1 Detailed Description

Represents an IP address.

Since

1.0

## 6.396.2 Constructor & Destructor Documentation

6.396.2.1 `decaf::net::InetAddress::InetAddress ( )` [protected]

6.396.2.2 `decaf::net::InetAddress::InetAddress ( const unsigned char * ipAddress, int numBytes )` [protected]

6.396.2.3 `decaf::net::InetAddress::InetAddress ( const std::string & hostname, const unsigned char * ipAddress, int numBytes )` [protected]

6.396.2.4 `virtual decaf::net::InetAddress::~~InetAddress ( )` [virtual]

## 6.396.3 Member Function Documentation

6.396.3.1 `static unsigned int decaf::net::InetAddress::bytesToInt ( const unsigned char * bytes, int start )` [static, protected]

Converts the bytes in an address array to an int starting from the value *start* treating the *start* value as the high order byte.

### Parameters

<i>bytes</i>	The array of bytes to convert to an int.
<i>start</i>	The index in the array to treat as the high order byte.

### Returns

an unsigned int that represents the address value.

6.396.3.2 `virtual decaf::lang::ArrayPointer<unsigned char> decaf::net::InetAddress::getAddress ( ) const` [virtual]

Returns the Raw IP address in Network byte order.

The returned address is a copy of the bytes contained in this **InetAddress** (p. 2077).

### Returns

and ArrayPointer containing the raw bytes of the network address.

6.396.3.3 `static InetAddress decaf::net::InetAddress::getAnyAddress ( )` [static, protected]

6.396.3.4 `static InetAddress decaf::net::InetAddress::getByAddress ( const std::string & hostname, const unsigned char * bytes, int numBytes )` [static]

Given a host name and IPAddress return a new **InetAddress** (p. 2077).

There is no name service checking or address validation done on the provided host name. The host name can either be machine name or the text based representation of the IP Address.

An IPV4 address must be only four bytes in length and an IPV6 address must be 16 bytes in length.

### Returns

a copy of an **InetAddress** (p. 2077) that represents the given byte array address.

### Exceptions

<b><i>UnknownHostException</i></b> (p. 4019)	if the address array length is invalid.
---	---

**6.396.3.5** `static InetAddress decaf::net::InetAddress::getByAddress ( const unsigned char * bytes, int numBytes ) [static]`

Given a raw IP Address in byte array form, create and return a new **InetAddress** (p. 2077) instance.

An IPV4 address must be only four bytes in length and an IPV6 address must be 16 bytes in length.

### Returns

a copy of an **InetAddress** (p. 2077) that represents the given byte array address.

### Exceptions

<b><i>UnknownHostException</i></b> (p. 4019)	if the address array length is invalid.
---	---

**6.396.3.6** `virtual std::string decaf::net::InetAddress::getHostAddress ( ) const [virtual]`

Returns a textual representation of the IP Address.

### Returns

the string form of the IP Address.

**6.396.3.7** `virtual std::string decaf::net::InetAddress::getHostName ( ) const [virtual]`

Get the host name associated with this **InetAddress** (p. 2077) instance.

If a host name was set upon construction then that value is returned, otherwise a reverse name lookup will be performed to attempt to get the host name associated with the set IP Address. If the host name cannot be resolved the textual representation of the IP Address is returned instead.

### Returns

the name of the host associated with this set IP Address.

#### 6.396.3.8 static `InetAddress decaf::net::InetAddress::getLocalHost ( )` [static]

Gets an **InetAddress** (p. 2077) that is the local host address.

If the localhost value cannot be resolved then the **InetAddress** (p. 2077) for Loopback is returned.

### Returns

a new **InetAddress** (p. 2077) object that contains the local host address.

### Exceptions

<i>UnknownHostException</i> (p. 4019)	if the address for local host is not found.
--	---

#### 6.396.3.9 static `InetAddress decaf::net::InetAddress::getLoopbackAddress ( )` [static, protected]

#### 6.396.3.10 virtual `bool decaf::net::InetAddress::isAnyLocalAddress ( ) const` [inline, virtual]

Check if this **InetAddress** (p. 2077) is a valid wildcard address.

### Returns

true if the address is a wildcard address.

Reimplemented in **decaf::net::Inet4Address** (p. 2073).

#### 6.396.3.11 virtual `bool decaf::net::InetAddress::isLinkLocalAddress ( ) const` [inline, virtual]

Check if this **InetAddress** (p. 2077) is a valid link local address.

### Returns

true if the address is a link local address.

Reimplemented in **decaf::net::Inet4Address** (p. 2074).

**6.396.3.12** `virtual bool decaf::net::InetAddress::isLoopbackAddress ( ) const [inline, virtual]`

Check if this **InetAddress** (p. 2077) is a valid loopback address.

#### Returns

true if the address is a loopback address.

Reimplemented in **decaf::net::Inet4Address** (p. 2074).

**6.396.3.13** `virtual bool decaf::net::InetAddress::isMCGlobal ( ) const [inline, virtual]`

Check if this **InetAddress** (p. 2077) is Multicast and has Global scope.

#### Returns

true if the address is Multicast and has Global scope.

Reimplemented in **decaf::net::Inet4Address** (p. 2074).

**6.396.3.14** `virtual bool decaf::net::InetAddress::isMCLinkLocal ( ) const [inline, virtual]`

Check if this **InetAddress** (p. 2077) is Multicast and has Link Local scope.

#### Returns

true if the address is Multicast and has Link Local scope.

Reimplemented in **decaf::net::Inet4Address** (p. 2074).

**6.396.3.15** `virtual bool decaf::net::InetAddress::isMCNodeLocal ( ) const [inline, virtual]`

Check if this **InetAddress** (p. 2077) is Multicast and has Node Local scope.

#### Returns

true if the address is Multicast and has Node Local scope.

Reimplemented in **decaf::net::Inet4Address** (p. 2075).

**6.396.3.16** `virtual bool decaf::net::InetAddress::isMCOrgLocal ( ) const [inline, virtual]`

Check if this **InetAddress** (p. 2077) is Multicast and has Organization scope.

**Returns**

true if the address is Multicast and has Organization scope.

Reimplemented in **decaf::net::Inet4Address** (p. 2075).

**6.396.3.17** `virtual bool decaf::net::InetAddress::isMCSiteLocal ( ) const [inline, virtual]`

Check if this **InetAddress** (p. 2077) is Multicast and has Site Local scope.

**Returns**

true if the address is Multicast and has Site Local scope.

Reimplemented in **decaf::net::Inet4Address** (p. 2075).

**6.396.3.18** `virtual bool decaf::net::InetAddress::isMulticastAddress ( ) const [inline, virtual]`

Check if this **InetAddress** (p. 2077) is a valid Multicast address.

**Returns**

true if the address is a Multicast address.

Reimplemented in **decaf::net::Inet4Address** (p. 2075).

**6.396.3.19** `virtual bool decaf::net::InetAddress::isSiteLocalAddress ( ) const [inline, virtual]`

Check if this **InetAddress** (p. 2077) is a valid site local address.

**Returns**

true if the address is a site local address.

Reimplemented in **decaf::net::Inet4Address** (p. 2075).

**6.396.3.20** `virtual std::string decaf::net::InetAddress::toString ( ) const [virtual]`

Returns a string representation of the **InetAddress** (p. 2077) in the form 'hostname / ipaddress'.

If the hostname is not resolved than it appears as empty.

**Returns**

string value of this **InetAddress** (p. 2077).

### 6.396.4 Field Documentation

- 6.396.4.1 **decaf::lang::ArrayPointer<unsigned char>**  
**decaf::net::InetSocketAddress::addressBytes** [mutable, protected]
- 6.396.4.2 **const unsigned char decaf::net::InetSocketAddress::anyBytes[4]** [static, protected]
- 6.396.4.3 **std::string decaf::net::InetSocketAddress::hostname** [mutable, protected]
- 6.396.4.4 **const unsigned char decaf::net::InetSocketAddress::loopbackBytes[4]**  
[static, protected]
- 6.396.4.5 **bool decaf::net::InetSocketAddress::reached** [mutable, protected]

The documentation for this class was generated from the following file:

- src/main/decaf/net/**InetSocketAddress.h**

## 6.397 decaf::net::InetSocketAddress Class Reference

```
#include <src/main/decaf/net/InetSocketAddress.h>
```

Inheritance diagram for decaf::net::InetSocketAddress:

### Public Member Functions

- **InetSocketAddress ()**
- virtual **~InetSocketAddress ()**

### 6.397.1 Constructor & Destructor Documentation

- 6.397.1.1 **decaf::net::InetSocketAddress::InetSocketAddress ( )**
- 6.397.1.2 **virtual decaf::net::InetSocketAddress::~~InetSocketAddress ( )** [virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/net/**InetSocketAddress.h**

## 6.398 inflate\_state Struct Reference

```
#include <src/main/decaf/internal/util/zip/inflate.h>
```

**Data Fields**

- **inflate\_mode** mode
- int **last**
- int **wrap**
- int **havedict**
- int **flags**
- unsigned **dmax**
- unsigned long **check**
- unsigned long **total**
- **gz\_headerp** head
- unsigned **wbits**
- unsigned **wsiz**
- unsigned **whave**
- unsigned **wnext**
- unsigned char FAR \* **window**
- unsigned long **hold**
- unsigned **bits**
- unsigned **length**
- unsigned **offset**
- unsigned **extra**
- **code** const FAR \* **lencode**
- **code** const FAR \* **distcode**
- unsigned **lenbits**
- unsigned **distbits**
- unsigned **ncode**
- unsigned **nlen**
- unsigned **ndist**
- unsigned **have**
- **code** FAR \* **next**
- unsigned short **lens** [320]
- unsigned short **work** [288]
- **code** **codes** [ENOUGH]
- int **sane**
- int **back**
- unsigned **was**



### 6.398.1 Field Documentation

- 6.398.1.1 `int inflate_state::back`
- 6.398.1.2 `unsigned inflate_state::bits`
- 6.398.1.3 `unsigned long inflate_state::check`
- 6.398.1.4 `code inflate_state::codes[ENOUGH]`
- 6.398.1.5 `unsigned inflate_state::distbits`
- 6.398.1.6 `code const FAR* inflate_state::distcode`
- 6.398.1.7 `unsigned inflate_state::dmax`
- 6.398.1.8 `unsigned inflate_state::extra`
- 6.398.1.9 `int inflate_state::flags`
- 6.398.1.10 `unsigned inflate_state::have`
- 6.398.1.11 `int inflate_state::havedict`
- 6.398.1.12 `gz_headerp inflate_state::head`
- 6.398.1.13 `unsigned long inflate_state::hold`
- 6.398.1.14 `int inflate_state::last`
- 6.398.1.15 `unsigned inflate_state::lenbits`
- 6.398.1.16 `code const FAR* inflate_state::lencode`
- 6.398.1.17 `unsigned inflate_state::length`
- 6.398.1.18 `unsigned short inflate_state::lens[320]`
- 6.398.1.19 `inflate_mode inflate_state::mode`
- 6.398.1.20 `unsigned inflate_state::ncode`
- 6.398.1.21 `unsigned inflate_state::ndist`
- 6.398.1.22 `code FAR* inflate_state::next`
- 6.398.1.23 `unsigned inflate_state::nlen`
- 6.398.1.24 `unsigned inflate_state::offset`
- 6.398.1.25 `int inflate_state::sane`
- 6.398.1.26 `unsigned long inflate_state::total`
- 6.398.1.27 `unsigned inflate_state::was`
- 6.398.1.28 `unsigned inflate_state::wbits`
- 6.398.1.29 `unsigned inflate_state::whave`

- `src/main/decaf/internal/util/zip/inflate.h`

## 6.399 decaf::util::zip::Inflater Class Reference

This class uncompresses data that was compressed using the *DEFLATE* algorithm (see *specification*).

```
#include <src/main/decaf/util/zip/Inflater.h>
```

### Public Member Functions

- **Inflater** ()

*Creates a new decompressor.*

- **Inflater** (bool nowrap)

*Creates a new decompressor.*

- virtual **~Inflater** ()

- void **setInput** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

*Sets input data for decompression.*

- void **setInput** (const std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

*Sets input data for decompression.*

- void **setInput** (const std::vector< unsigned char > &buffer) throw ( decaf::lang::exceptions::IllegalStateException )

*Sets input data for decompression.*

- int **getRemaining** () const

*Returns the total number of bytes remaining in the input buffer.*

- void **setDictionary** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Sets the preset dictionary to the given array of bytes.*

- void **setDictionary** (const std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Sets the preset dictionary to the given array of bytes.*

- void **setDictionary** (const std::vector< unsigned char > &buffer) throw ( decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IllegalArgumentException )  
*Sets the preset dictionary to the given array of bytes.*
- bool **needsInput** () const
- bool **needsDictionary** () const
- void **finish** ()  
*When called, indicates that decompression should end with the current contents of the input buffer.*
- bool **finished** () const
- int **inflate** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::util::zip::DataFormatException )  
*Uncompresses bytes into specified buffer.*
- int **inflate** (std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::util::zip::DataFormatException )  
*Uncompresses bytes into specified buffer.*
- int **inflate** (std::vector< unsigned char > &buffer) throw ( decaf::lang::exceptions::IllegalStateException, decaf::util::zip::DataFormatException )  
*Uncompresses bytes into specified buffer.*
- long long **getAdler** () const throw ( decaf::lang::exceptions::IllegalStateException )
- long long **getBytesRead** () const throw ( decaf::lang::exceptions::IllegalStateException )
- long long **getBytesWritten** () const throw ( decaf::lang::exceptions::IllegalStateException )
- void **reset** () throw ( decaf::lang::exceptions::IllegalStateException )  
*Resets deflater so that a new set of input data can be processed.*
- void **end** ()  
*Closes the decompressor and discards any unprocessed input.*

### 6.399.1 Detailed Description

This class uncompresses data that was compressed using the *DEFLATE* algorithm (see specification). Basically this class is part of the API to the stream based ZLIB compression library and is used as such by **InflaterInputStream** (p. 2097) and its descendants.

The typical usage of a **Inflater** (p. 2088) outside this package consists of a specific call to one of its constructors before being passed to an instance of **InflaterInputStream** (p. 2097).

#### See also

**InflaterInputStream** (p. 2097)

**Deflater** (p. 1759)

#### Since

1.0

### 6.399.2 Constructor & Destructor Documentation

#### 6.399.2.1 `decaf::util::zip::Inflater::Inflater ( )`

Creates a new decompressor.

This constructor defaults the inflater to use the ZLIB header and checksum fields.

#### 6.399.2.2 `decaf::util::zip::Inflater::Inflater ( bool nowrap )`

Creates a new decompressor.

If the parameter 'nowrap' is true then the ZLIB header and checksum fields will not be used. This provides compatibility with the compression format used by both GZIP and PKZIP.

Note: When using the 'nowrap' option it is also necessary to provide an extra "dummy" byte as input. This is required by the ZLIB native library in order to support certain optimizations.

#### 6.399.2.3 `virtual decaf::util::zip::Inflater::~~Inflater ( )` [virtual]

### 6.399.3 Member Function Documentation

#### 6.399.3.1 `void decaf::util::zip::Inflater::end ( )`

Closes the decompressor and discards any unprocessed input.

This method should be called when the decompressor is no longer being used, but will also be called automatically by the destructor. Once this method is called, the behavior of the **Inflater** (p. 2088) object is undefined.

#### 6.399.3.2 `void decaf::util::zip::Inflater::finish ( )`

When called, indicates that decompression should end with the current contents of the input buffer.

**6.399.3.3** `bool decaf::util::zip::Inflater::finished ( ) const`

#### Returns

true if the end of the compressed data output stream has been reached.

**6.399.3.4** `long long decaf::util::zip::Inflater::getAdler ( ) const throw ( decaf::lang::exceptions::IllegalStateException )`

#### Returns

the ADLER-32 value of the uncompressed data.

#### Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

**6.399.3.5** `long long decaf::util::zip::Inflater::getBytesRead ( ) const throw ( decaf::lang::exceptions::IllegalStateException )`

#### Returns

the total number of compressed bytes input so far.

#### Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

**6.399.3.6** `long long decaf::util::zip::Inflater::getBytesWritten ( ) const throw ( decaf::lang::exceptions::IllegalStateException )`

#### Returns

the total number of decompressed bytes output so far.

#### Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

**6.399.3.7** `int decaf::util::zip::Inflater::getRemaining ( ) const`

Returns the total number of bytes remaining in the input buffer.

This can be used to find out what bytes still remain in the input buffer after decompression has finished.

**Returns**

the total number of bytes remaining in the input buffer

**6.399.3.8** `int decaf::util::zip::Inflater::inflate ( std::vector< unsigned char > & buffer ) throw ( decaf::lang::exceptions::IllegalStateException, decaf::util::zip::DataFormatException )`

Uncompresses bytes into specified buffer.

Returns actual number of bytes uncompressed. A return value of 0 indicates that **needsInput()** (p. 2093) or **needsDictionary()** (p. 2093) should be called in order to determine if more input data or a preset dictionary is required. In the latter case, **getAdler()** (p. 2091) can be used to get the Adler-32 value of the dictionary required.

**Parameters**

<i>buffer</i>	The Buffer to write the compressed data to.
---------------	---

**Exceptions**

<i>IllegalStateException</i>	if in the end state.
<i><b>DataFormatException</b></i> (p. 1600)	if the compressed data format is invalid.

**6.399.3.9** `int decaf::util::zip::Inflater::inflate ( std::vector< unsigned char > & buffer, int offset, int length ) throw ( decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::util::zip::DataFormatException )`

Uncompresses bytes into specified buffer.

Returns actual number of bytes uncompressed. A return value of 0 indicates that **needsInput()** (p. 2093) or **needsDictionary()** (p. 2093) should be called in order to determine if more input data or a preset dictionary is required. In the latter case, **getAdler()** (p. 2091) can be used to get the Adler-32 value of the dictionary required.

**Parameters**

<i>buffer</i>	The Buffer to write the compressed data to.
<i>offset</i>	The position in the Buffer to start writing at.
<i>length</i>	The maximum number of byte of data to write.

**Exceptions**

<i>IllegalStateException</i>	if in the end state.
<i><b>IndexOutOfBoundsException</b></i>	if the offset + length > size of the buffer.

<b><i>DataFormatException</i></b> (p. 1600)	if the compressed data format is invalid.
--	---

**6.399.3.10** int decaf::util::zip::Inflater::inflate ( unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::util::zip::DataFormatException )

Uncompresses bytes into specified buffer.

Returns actual number of bytes uncompressed. A return value of 0 indicates that **needsInput()** (p. 2093) or **needsDictionary()** (p. 2093) should be called in order to determine if more input data or a preset dictionary is required. In the latter case, **getAdler()** (p. 2091) can be used to get the Adler-32 value of the dictionary required.

#### Parameters

<i>buffer</i>	The Buffer to write the compressed data to.
<i>size</i>	The size of the buffer passed in.
<i>offset</i>	The position in the Buffer to start writing at.
<i>length</i>	The maximum number of byte of data to write.

#### Exceptions

<b><i>NullPointerException</i></b>	if buffer is NULL.
<b><i>IllegalStateException</i></b>	if in the end state.
<b><i>IndexOutOfBoundsException</i></b>	if the offset + length > size of the buffer.
<b><i>DataFormatException</i></b> (p. 1600)	if the compressed data format is invalid.

**6.399.3.11** bool decaf::util::zip::Inflater::needsDictionary ( ) const

#### Returns

true if a preset dictionary is needed for decompression.

**6.399.3.12** bool decaf::util::zip::Inflater::needsInput ( ) const

#### Returns

true if the input data buffer is empty and **setInput()** (p. 2095) should be called in order to provide more input

**6.399.3.13** void `decaf::util::zip::Inflater::reset ( )` throw ( `decaf::lang::exceptions::IllegalStateException` )

Resets deflater so that a new set of input data can be processed.

Keeps current decompression level and strategy settings.

#### Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

**6.399.3.14** void `decaf::util::zip::Inflater::setDictionary ( const std::vector< unsigned char > & buffer, int offset, int length )` throw ( `decaf::lang::exceptions::IndexOutOfBoundsException`, `decaf::lang::exceptions::IllegalArgumentException`, `decaf::lang::exceptions::IllegalStateException` )

Sets the preset dictionary to the given array of bytes.

Should be called when **inflate()** (p. 2093) returns 0 and **needsDictionary()** (p. 2093) returns true indicating that a preset dictionary is required. The method **getAdler()** (p. 2091) can be used to get the Adler-32 value of the dictionary needed.

#### Parameters

<i>buffer</i>	The Buffer to read in for decompression.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.
<i>IllegalArgumentException</i>	if the given dictionary doesn't match the required dictionaries checksum value.

**6.399.3.15** void `decaf::util::zip::Inflater::setDictionary ( const unsigned char * buffer, int size, int offset, int length )` throw ( `decaf::lang::exceptions::NullPointerException`, `decaf::lang::exceptions::IndexOutOfBoundsException`, `decaf::lang::exceptions::IllegalArgumentException`, `decaf::lang::exceptions::IllegalStateException` )

Sets the preset dictionary to the given array of bytes.

Should be called when **inflate()** (p. 2093) returns 0 and **needsDictionary()** (p. 2093) returns true indicating that a preset dictionary is required. The method **getAdler()** (p. 2091) can be used to get the Adler-32 value of the dictionary needed.



**Parameters**

<i>buffer</i>	The Buffer to read in for decompression.
<i>size</i>	The size of the buffer passed in.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

**Exceptions**

<i>NullPointerException</i>	if buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.
<i>IllegalArgumentException</i>	if the given dictionary doesn't match the required dictionaries checksum value.

**6.399.3.16** void decaf::util::zip::Inflater::setDictionary ( const std::vector< unsigned char > & *buffer* ) throw ( decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IllegalArgumentException )

Sets the preset dictionary to the given array of bytes.

Should be called when **inflate()** (p. 2093) returns 0 and **needsDictionary()** (p. 2093) returns true indicating that a preset dictionary is required. The method **getAdler()** (p. 2091) can be used to get the Adler-32 value of the dictionary needed.

**Parameters**

<i>buffer</i>	The Buffer to read in for decompression.
---------------	--

**Exceptions**

<i>IllegalStateException</i>	if in the end state.
<i>IllegalArgumentException</i>	if the given dictionary doesn't match the required dictionaries checksum value.

**6.399.3.17** void decaf::util::zip::Inflater::setInput ( const unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

Sets input data for decompression.

This should be called whenever **needsInput()** (p. 2093) returns true indicating that more input data is required.

**Parameters**

<i>buffer</i>	The Buffer to read in for decompression.
---------------	--

<i>size</i>	The size of the buffer passed in.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

**Exceptions**

<i>NullPointerException</i>	if buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

**6.399.3.18** `void decaf::util::zip::Inflater::setInput ( const std::vector< unsigned char > & buffer, int offset, int length ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )`

Sets input data for decompression.

This should be called whenever **needsInput()** (p. 2093) returns true indicating that more input data is required.

**Parameters**

<i>buffer</i>	The Buffer to read in for decompression.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

**6.399.3.19** `void decaf::util::zip::Inflater::setInput ( const std::vector< unsigned char > & buffer ) throw ( decaf::lang::exceptions::IllegalStateException )`

Sets input data for decompression.

This should be called whenever **needsInput()** (p. 2093) returns true indicating that more input data is required.

**Parameters**

<i>buffer</i>	The Buffer to read in for decompression.
---------------	--

**Exceptions**

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**Inflater.h**

**6.400 decaf::util::zip::InflaterInputStream Class Reference**

A `FilterInputStream` that decompresses data read from the wrapped `InputStream` instance.

```
#include <src/main/decaf/util/zip/InflaterInputStream.h>
```

Inheritance diagram for `decaf::util::zip::InflaterInputStream`:

**Public Member Functions**

- **InflaterInputStream** (`decaf::io::InputStream *inputStream`, `bool own=false`)

*Create an instance of this class with a default inflater and buffer size.*

- **InflaterInputStream** (`decaf::io::InputStream *inputStream`, `Inflater *inflater`, `bool own=false`)

*Creates a new **InflaterInputStream** (p. 2097) with a user supplied **Inflater** (p. 2088) and a default buffer size.*

- **InflaterInputStream** (`decaf::io::InputStream *inputStream`, `Inflater *inflater`, `int bufferSize`, `bool own=false`)

*Creates a new **DeflateOutputStream** with a user supplied **Inflater** (p. 2088) and specified buffer size.*

- virtual **~InflaterInputStream** ()
- virtual `int available` () const throw ( `decaf::io::IOException` )

*Indicates the number of bytes available.*

*The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.*

*The default implementation of this method returns zero.*

**Returns**

*the number of bytes available on this input stream.*

**Exceptions**

<b>IOException</b> (p. 2210)	if an I/O error occurs.
------------------------------	-------------------------

- virtual void **close** () throw ( decaf::io::IOException )

*Closes the **InputStream** (p. 2105) freeing any resources that might have been aquired during the lifetime of this stream.*

*The default implementation of this method does nothing.*

- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Skips over and discards n bytes of data from this input stream.*

*The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.*

*The skip method of **InputStream** (p. 2105) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*

#### Parameters

num	The number of bytes to skip.
-----	------------------------------

#### Returns

*total bytes skipped*

#### Exceptions

<b>IOException</b> (p. 2210)	<i>if an I/O error occurs.</i>
UnsupportedOperationException	<i>if the concrete stream class does not support skipping bytes.</i>

- virtual void **mark** (int readLimit)

*Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.*

*If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.*

*Calling mark on a closed stream instance should have no effect.*

*The default implementation of this method does nothing.*

#### Parameters

readLimit	The max bytes read before marked position is invalid.
-----------	---

- virtual void **reset** () throw ( decaf::io::IOException )

*Repositions this stream to the position at the time the mark method was last called on this input stream.*

*If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2210) might be thrown. \* If such an **IOException** (p. 2210) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that*

otherwise would have been the next input data as of the time of the call to reset.

If the method `markSupported` returns false, then: \* The call to reset may throw an **IOException** (p. 2210). \* If an **IOException** (p. 2210) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the `read` method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2210).

#### Exceptions

<b>IOException</b> (p. 2210)	if an I/O error occurs.
------------------------------	-------------------------

- virtual bool **markSupported** () const

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

#### Returns

true if this stream instance supports marks

### Protected Member Functions

- virtual void **fill** () throw ( decaf::io::IOException )  
Fills the input buffer with the next chunk of data.
- virtual int **doReadByte** () throw ( decaf::io::IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int **length**) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

### Protected Attributes

- **Inflater \* inflater**  
The **Inflater** (p. 2088) instance to use.
- std::vector< unsigned char > **buff**  
The buffer to hold chunks of data read from the stream before inflation.
- int **length**  
The amount of data currently stored in the input buffer.
- bool **ownInflater**
- bool **atEOF**

### Static Protected Attributes

- static const int **DEFAULT\_BUFFER\_SIZE**

### 6.400.1 Detailed Description

A `FilterInputStream` that decompresses data read from the wrapped `InputStream` instance.

Since

1.0

### 6.400.2 Constructor & Destructor Documentation

**6.400.2.1** `decaf::util::zip::InflaterInputStream::InflaterInputStream ( decaf::io::InputStream * inputStream, bool own = false )`

Create an instance of this class with a default inflater and buffer size.

#### Parameters

<i>inputStream</i>	The <code>InputStream</code> instance to wrap.
<i>own</i>	Should this <code>Filter</code> take ownership of the <code>InputStream</code> pointer (defaults to false).

**6.400.2.2** `decaf::util::zip::InflaterInputStream::InflaterInputStream ( decaf::io::InputStream * inputStream, Inflater * inflater, bool own = false )`

Creates a new `InflaterInputStream` (p. 2097) with a user supplied `Inflater` (p. 2088) and a default buffer size.

When the user supplied a `Inflater` (p. 2088) instance the `InflaterInputStream` (p. 2097) does not take ownership of the `Inflater` (p. 2088) pointer, the caller is still responsible for deleting the `Inflater` (p. 2088).

#### Parameters

<i>inputStream</i>	The <code>InputStream</code> instance to wrap.
<i>inflater</i>	The user supplied <code>Inflater</code> (p. 2088) to use for decompression. (
<i>own</i>	Should this filter take ownership of the <code>InputStream</code> pointer (default is false).

#### Exceptions

<i>NullPointerException</i>	if the <code>Inflater</code> (p. 2088) given is <code>NULL</code> .
-----------------------------	---

**6.400.2.3** `decaf::util::zip::InflaterInputStream::InflaterInputStream ( decaf::io::InputStream * inputStream, Inflater * inflater, int bufferSize, bool own = false )`

Creates a new DeflateOutputStream with a user supplied **Inflater** (p. 2088) and specified buffer size.

When the user supplied a **Inflater** (p. 2088) instance the **InflaterInputStream** (p. 2097) does not take ownership of the **Inflater** (p. 2088) pointer, the caller is still responsible for deleting the **Inflater** (p. 2088).

#### Parameters

<i>inputStream</i>	The InputStream instance to wrap.
<i>inflater</i>	The user supplied <b>Inflater</b> (p. 2088) to use for decompression.
<i>bufferSize</i>	The size of the input buffer.
<i>own</i>	Should this filter take ownership of the InputStream pointer (default is false).

#### Exceptions

<i>NullPointerException</i>	if the <b>Inflater</b> (p. 2088) given is NULL.
<i>IllegalArgumentException</i>	if the bufferSize value is zero.

**6.400.2.4** `virtual decaf::util::zip::InflaterInputStream::~InflaterInputStream ( ) [virtual]`

### 6.400.3 Member Function Documentation

**6.400.3.1** `virtual int decaf::util::zip::InflaterInputStream::available ( ) const throw ( decaf::io::IOException ) [virtual]`

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

#### Returns

the number of bytes available on this input stream.

#### Exceptions

<i>IOException</i> (p. 2210)	if an I/O error occurs.
---------------------------------	-------------------------

Until EOF this method always returns 1, thereafter it always returns 0.

Reimplemented from **decaf::io::FilterInputStream** (p. 1953).

**6.400.3.2** `virtual void decaf::util::zip::InflaterInputStream::close ( ) throw ( decaf::io::IOException )` [virtual]

Closes the **InputStream** (p. 2105) freeing any resources that might have been aquired during the lifetime of this stream.

The default implementation of this method does nothing.

Closes any resources associated with this **InflaterInputStream** (p. 2097).

Reimplemented from **decaf::io::FilterInputStream** (p. 1954).

**6.400.3.3** `virtual int decaf::util::zip::InflaterInputStream::doReadArrayBounded ( unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )` [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1954).

**6.400.3.4** `virtual int decaf::util::zip::InflaterInputStream::doReadByte ( ) throw ( decaf::io::IOException )` [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1954).

**6.400.3.5** `virtual void decaf::util::zip::InflaterInputStream::fill ( ) throw ( decaf::io::IOException )` [protected, virtual]

Fills the input buffer with the next chunk of data.

### Exceptions

<i>IOException</i> if an I/O error occurs.
--

**6.400.3.6** `virtual void decaf::util::zip::InflaterInputStream::mark ( int readLimit )` [virtual]

Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.



The default implementation of this method does nothing.

#### Parameters

<i>readLimit</i>	The max bytes read before marked position is invalid.
------------------	---

Does nothing.

Reimplemented from **decaf::io::FilterInputStream** (p. 1955).

**6.400.3.7** `virtual bool decaf::util::zip::InflaterInputStream::markSupported ( ) const`  
[virtual]

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

#### Returns

true if this stream instance supports marks

Always returns false.

Reimplemented from **decaf::io::FilterInputStream** (p. 1955).

**6.400.3.8** `virtual void decaf::util::zip::InflaterInputStream::reset ( ) throw (`  
`decaf::io::IOException )` [virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2210) might be thrown. \* If such an **IOException** (p. 2210) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2210). \* If an **IOException** (p. 2210) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2210).

#### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
--	-------------------------

Always throws an `IOException` when called.

Reimplemented from `decaf::io::FilterInputStream` (p. 1956).

**6.400.3.9** `virtual long long decaf::util::zip::InflaterInputStream::skip  
( long long num ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of `InputStream` (p. 2105) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

#### Returns

total bytes skipped

#### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
<i>UnsupportedOperation</i> <i>Exception</i>	if the concrete stream class does not support skipping bytes.

Skips the specified amount of uncompressed input data.

Reimplemented from `decaf::io::FilterInputStream` (p. 1956).

### 6.400.4 Field Documentation

**6.400.4.1** `bool decaf::util::zip::InflaterInputStream::atEOF` [protected]

**6.400.4.2** `std::vector<unsigned char> decaf::util::zip::InflaterInputStream::buff`  
[protected]

The buffer to hold chunks of data read from the stream before inflation.

**6.400.4.3** `const int decaf::util::zip::InflaterInputStream::DEFAULT_BUFFER_SIZE` [static, protected]

**6.400.4.4** `Inflater* decaf::util::zip::InflaterInputStream::inflater` [protected]

The **Inflater** (p. 2088) instance to use.

**6.400.4.5** `int decaf::util::zip::InflaterInputStream::length` [protected]

The amount of data currently stored in the input buffer.

**6.400.4.6** `bool decaf::util::zip::InflaterInputStream::ownInflater` [protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/InflaterInputStream.h`

## 6.401 decaf::io::InputStream Class Reference

A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.

```
#include <src/main/decaf/io/InputStream.h>
```

Inheritance diagram for decaf::io::InputStream:

### Public Member Functions

- **InputStream** ()
- virtual **~InputStream** ()
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes the **InputStream** (p. 2105) freeing any resources that might have been acquired during the lifetime of this stream.*
- virtual void **mark** (int readLimit)  
*Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.*
- virtual void **reset** () throw ( decaf::io::IOException )  
*Repositions this stream to the position at the time the mark method was last called on this input stream.*

- virtual bool **markSupported** () const  
*Determines if this input stream supports the mark and reset methods.*
- virtual int **available** () const throw ( decaf::io::IOException )  
*Indicates the number of bytes available.*
- virtual int **read** () throw ( decaf::io::IOException )  
*Reads a single byte from the buffer.*
- virtual int **read** (unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )  
*Reads up to size bytes of data from the input stream into an array of bytes.*
- virtual int **read** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Reads up to length bytes of data from the input stream into an array of bytes.*
- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Skips over and discards n bytes of data from this input stream.*
- virtual std::string **toString** () const  
*Output a String representation of this object.*
- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Locks the object.*
- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Attempts to Lock the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Unlocks the object.*
- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals a waiter on this object that it can now wake up and continue.*

- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals the waiters on this object that it can now wake up and continue.*

### Protected Member Functions

- virtual int **doReadByte** ()=0 throw ( decaf::io::IOException )
- virtual int **doReadArray** (unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

#### 6.401.1 Detailed Description

A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.

Since

1.0

#### 6.401.2 Constructor & Destructor Documentation

6.401.2.1 decaf::io::InputStream::InputStream ( )

6.401.2.2 virtual decaf::io::InputStream::~~InputStream ( ) [virtual]

#### 6.401.3 Member Function Documentation

6.401.3.1 virtual int decaf::io::InputStream::available ( ) const throw ( decaf::io::IOException ) [inline, virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

**Returns**

the number of bytes available on this input stream.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
--	-------------------------

Reimplemented in **decaf::internal::io::StandardInputStream** (p. 3689), **decaf::internal::net::ssl::openssl::OpenSSLInputStream** (p. 2967), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 3861), **decaf::io::BlockingByteArrayInputStream** (p. 847), **decaf::io::BufferedInputStream** (p. 946), **decaf::io::ByteArrayInputStream** (p. 1043), **decaf::io::FilterInputStream** (p. 1953), **decaf::io::PushbackInputStream** (p. 3234), and **decaf::util::zip::InflaterInputStream** (p. 2101).

**6.401.3.2** `virtual void decaf::io::InputStream::close ( ) throw ( decaf::io::IOException )`  
[virtual]

Closes the **InputStream** (p. 2105) freeing any resources that might have been aquired during the lifetime of this stream.

The default implementation of this method does nothing.

Implements **decaf::io::Closeable** (p. 1181).

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream** (p. 2968), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 3862), **decaf::io::BlockingByteArrayInputStream** (p. 847), **decaf::io::BufferedInputStream** (p. 946), **decaf::io::FilterInputStream** (p. 1954), and **decaf::util::zip::InflaterInputStream** (p. 2102).

**6.401.3.3** `virtual int decaf::io::InputStream::doReadArray ( unsigned char * buffer, int size ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )` [protected, virtual]

Reimplemented in **decaf::io::FilterInputStream** (p. 1954).

**6.401.3.4** `virtual int decaf::io::InputStream::doReadArrayBounded ( unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )` [protected, virtual]

Reimplemented in **activemq::io::LoggingInputStream** (p. 2475), **decaf::internal::net::ssl::openssl::OpenSSLInputStream** (p. 2968), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 3862), **decaf::io::BlockingByteArrayInputStream** (p. 847), **decaf::io::BufferedInputStream** (p. 946), **decaf::io::ByteArrayInputStream** (p. 1043), **decaf::io::FilterInputStream** (p. 1954), **decaf::io::PushbackInputStream** (p. 3235), **decaf::util::zip::CheckedInputStream** (p. 1171), and **decaf::util::zip::InflaterInputStream** (p. 2102).

**6.401.3.5** `virtual int decaf::io::InputStream::doReadByte ( ) throw ( decaf::io::IOException )` [protected, pure virtual]

Implemented in **activemq::io::LoggingInputStream** (p. 2475), **decaf::internal::io::StandardInputStream** (p. 3689), **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream** (p. 2968), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 3862), **decaf::io::BlockingByteArrayInputStream** (p. 847), **decaf::io::BufferedInputStream** (p. 946), **decaf::io::ByteArrayInputStream** (p. 1043), **decaf::io::FilterInputStream** (p. 1954), **decaf::io::PushbackInputStream** (p. 3235), **decaf::util::zip::CheckedInputStream** (p. 1171), and **decaf::util::zip::InflaterInputStream** (p. 2102).

**6.401.3.6** `virtual void decaf::io::InputStream::lock ( ) throw ( decaf::lang::exceptions::RuntimeException )` [inline, virtual]

Locks the object.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3812).

**6.401.3.7** `virtual void decaf::io::InputStream::mark ( int readLimit )` [virtual]

Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

#### Parameters

<i>readLimit</i>	The max bytes read before marked position is invalid.
------------------	---

Reimplemented in **decaf::io::BufferedInputStream** (p. 947), **decaf::io::ByteArrayInputStream** (p. 1043), **decaf::io::FilterInputStream** (p. 1955), **decaf::io::PushbackInputStream** (p. 3235), and **decaf::util::zip::InflaterInputStream** (p. 2102).

**6.401.3.8** `virtual bool decaf::io::InputStream::markSupported ( ) const` [inline, virtual]

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular

input stream instance.

The default implementation of this method returns false.

### Returns

true if this stream instance supports marks

Reimplemented in **decaf::io::BufferedInputStream** (p. 947), **decaf::io::ByteArrayInputStream** (p. 1044), **decaf::io::FilterInputStream** (p. 1955), **decaf::io::PushbackInputStream** (p. 3235), and **decaf::util::zip::InflaterInputStream** (p. 2103).

**6.401.3.9** `virtual void decaf::io::InputStream::notify ( ) throw  
( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalMonitorStateException ) [inline,  
virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3813).

**6.401.3.10** `virtual void decaf::io::InputStream::notifyAll ( ) throw  
( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalMonitorStateException ) [inline,  
virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3814).



**6.401.3.11** `virtual int decaf::io::InputStream::read ( unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException ) [virtual]`

Reads up to length bytes of data from the input stream into an array of bytes.

An attempt is made to read as many as length bytes, but a smaller number may be read. The number of bytes actually read is returned as an integer.

This method blocks until input data is available, end of file is detected, or an exception is thrown.

If length is zero, then no bytes are read and 0 is returned; otherwise, there is an attempt to read at least one byte. If no byte is available because the stream is at end of file, the value -1 is returned; otherwise, at least one byte is read and stored into b.

The first byte read is stored into element b[off], the next one into b[off+1], and so on. The number of bytes read is, at most, equal to length. Let k be the number of bytes actually read; these bytes will be stored in elements b[off] through b[off+k-1], leaving elements b[off+k] through b[off+length-1] unaffected.

In every case, elements b[0] through b[offset] and elements b[offset+length] through b[b.length-1] are unaffected.

This method called the protected virtual method doReadArrayBounded which simply calls the method **doReadByte()** (p. 2109) repeatedly. If the first such call results in an **IOException** (p. 2210), that exception is returned. If any subsequent call to **doReadByte()** (p. 2109) results in a **IOException** (p. 2210), the exception is caught and treated as if it were end of file; the bytes read up to that point are stored into the buffer and the number of bytes read before the exception occurred is returned. The default implementation of this method blocks until the requested amount of input data has been read, end of file is detected, or an exception is thrown. Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters

<i>buffer</i>	The target buffer to write the read in data to.
<i>size</i>	The size in bytes of the target buffer.
<i>offset</i>	The position in the buffer to start inserting the read in data.
<i>length</i>	The maximum number of bytes that should be read into buffer.

#### Returns

The number of bytes read or -1 if EOF is detected

#### Exceptions

<b>IOException</b> (p. 2210)	if an I/O error occurs.
<i>NullPointerException</i>	if buffer passed is NULL.
<i>IndexOutOfBoundsException</i>	if length > size - offset.

**6.401.3.12** `virtual int decaf::io::InputStream::read ( ) throw ( decaf::io::IOException )`  
[virtual]

Reads a single byte from the buffer.

The value byte is returned as an int in the range 0 to 255. If no byte is available because the end of the stream has been reached, the value -1 is returned. This method blocks until input data is available, the end of the stream is detected, or an exception is thrown.

The default implementation of this method calls the internal virtual method doReadByte which is a pure virtual method and must be overridden by all subclasses.

### Returns

The next byte or -1 if the end of stream is reached.

### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
--	-------------------------

**6.401.3.13** `virtual int decaf::io::InputStream::read ( unsigned char * buffer, int size ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )` [virtual]

Reads up to size bytes of data from the input stream into an array of bytes.

An attempt is made to read as many as size bytes, but a smaller number may be read. The number of bytes actually read is returned as an integer.

This method blocks until input data is available, end of file is detected, or an exception is thrown.

If size is zero, then no bytes are read and 0 is returned; otherwise, there is an attempt to read at least one byte. If no byte is available because the stream is at end of file, the value -1 is returned; otherwise, at least one byte is read and stored into b.

This method called the protected virtual method doReadArray which by default is the same as calling read( buffer, size, 0, size ). Subclasses can customize the behavior of this method by overriding the doReadArray method to provide a better performing read operation.

### Parameters

<i>buffer</i>	The target buffer to write the read in data to.
<i>size</i>	The size in bytes of the target buffer.

### Returns

The number of bytes read or -1 if EOF is detected

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
<b><i>NullPointerException</i></b>	if buffer passed is NULL.

**6.401.3.14** virtual void decaf::io::InputStream::reset ( ) throw ( decaf::io::IOException )  
[virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2210) might be thrown. \* If such an **IOException** (p. 2210) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2210). \* If an **IOException** (p. 2210) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2210).

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
--	-------------------------

Reimplemented in **decaf::io::BufferedInputStream** (p. 947), **decaf::io::ByteArrayInputStream** (p. 1044), **decaf::io::FilterInputStream** (p. 1956), **decaf::io::PushbackInputStream** (p. 3236), and **decaf::util::zip::InflaterInputStream** (p. 2103).

**6.401.3.15** virtual long long decaf::io::InputStream::skip ( long  
long num ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2105) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

#### Returns

total bytes skipped

#### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
<b><i>UnsupportedOperationException</i></b>	if the concrete stream class does not support skipping bytes.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream** (p. 2968), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 3862), **decaf::io::BlockingByteArrayInputStream** (p. 848), **decaf::io::BufferedInputStream** (p. 948), **decaf::io::ByteArrayInputStream** (p. 1046), **decaf::io::FilterInputStream** (p. 1956), **decaf::io::PushbackInputStream** (p. 3236), **decaf::util::zip::CheckedInputStream** (p. 1171), and **decaf::util::zip::InflaterInputStream** (p. 2104).

**6.401.3.16** `virtual std::string decaf::io::InputStream::toString ( ) const` [virtual]

Output a String representation of this object.

The default version of this method just prints the Class Name.

#### Returns

a string representation of the object.

**6.401.3.17** `virtual bool decaf::io::InputStream::tryLock ( ) throw ( decaf::lang::exceptions::RuntimeException )` [inline, virtual]

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

#### Returns

true if the lock was acquired, false if it is already held by another thread.

#### Exceptions

<b><i>RuntimeException</i></b>	if an error occurs while locking the object.
--------------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3815).

**6.401.3.18** `virtual void decaf::io::InputStream::unlock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Unlocks the object.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3816).

**6.401.3.19** `virtual void decaf::io::InputStream::wait ( long long milliseconds ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

#### Parameters

<i>milliseconds</i>	the time in milliseconds to wait, or WAIT_INFINITE
---------------------	--

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3819).

**6.401.3.20** `virtual void decaf::io::InputStream::wait ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
-------------------------	---

<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p.3818).

```
6.401.3.21 virtual void decaf::io::InputStream::wait ( long long millisecs, int
nanos ) throw ( decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException ) [inline,
virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

#### Exceptions

<i>IllegalArgumentException</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p.3820).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**InputStream.h**

## 6.402 decaf::io::InputStreamReader Class Reference

An **InputStreamReader** (p.2116) is a bridge from byte streams to character streams.

```
#include <src/main/decaf/io/InputStreamReader.h>
```

Inheritance diagram for decaf::io::InputStreamReader:

### Public Member Functions

- **InputStreamReader** (**InputStream** \*stream, bool own=false)  
*Create a new **InputStreamReader** (p. 2116) that wraps the given **InputStream** (p. 2105).*
- virtual ~**InputStreamReader** ()
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes this object and deallocates the appropriate resources.*
- virtual bool **ready** () const throw ( decaf::io::IOException )  
*Tells whether this stream is ready to be read.*

### Protected Member Functions

- virtual int **doReadArrayBounded** (char \*buffer, int size, int offset, int length)  
throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::NullPointerException )  
*Override this method to customize the functionality of the method read( unsigned char\* buffer, int size, int offset, int length ).*
- virtual void **checkClosed** () const throw ( decaf::io::IOException )

#### 6.402.1 Detailed Description

An **InputStreamReader** (p. 2116) is a bridge from byte streams to character streams. For top efficiency, consider wrapping an **InputStreamReader** (p. 2116) within a **BufferedReader**. For example:

```
BufferedReader* in = new BufferedReader( new InputStreamReader (p. 2116)( System.in, false ), true );
```

See also

**OutputStreamWriter** (p. 2999)

Since

1.0

## 6.402.2 Constructor & Destructor Documentation

6.402.2.1 `decaf::io::InputStreamReader::InputStreamReader ( InputStream * stream, bool own = false )`

Create a new **InputStreamReader** (p. 2116) that wraps the given **InputStream** (p. 2105).

### Parameters

<i>stream</i>	The <b>InputStream</b> (p. 2105) to read from. (cannot be null).
<i>own</i>	Does this object own the passed <b>InputStream</b> (p. 2105) (defaults to false).

### Exceptions

<i>NullPointerException</i>	if the passed <b>InputStream</b> (p. 2105) is NULL.
-----------------------------	---

6.402.2.2 `virtual decaf::io::InputStreamReader::~~InputStreamReader ( ) [virtual]`

## 6.402.3 Member Function Documentation

6.402.3.1 `virtual void decaf::io::InputStreamReader::checkClosed ( ) const throw ( decaf::io::IOException ) [protected, virtual]`

6.402.3.2 `virtual void decaf::io::InputStreamReader::close ( ) throw ( decaf::io::IOException ) [virtual]`

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

### Exceptions

<i>IOException</i> (p. 2210)	if an error occurs while closing.
---------------------------------	-----------------------------------

Implements **decaf::io::Closeable** (p. 1181).

6.402.3.3 `virtual int decaf::io::InputStreamReader::doReadArrayBounded ( char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException ) [protected, virtual]`

Override this method to customize the functionality of the method read( unsigned char\* buffer, int size, int offset, int length ).

All subclasses must override this method to provide the basic **Reader** (p. 3254) functionality.



Implements **decaf::io::Reader** (p. 3256).

**6.402.3.4** virtual bool decaf::io::InputStreamReader::ready ( ) const throw ( decaf::io::IOException ) [virtual]

Tells whether this stream is ready to be read.

#### Returns

True if the next **read()** (p. 3257) is guaranteed not to block for input, false otherwise. Note that returning false does not guarantee that the next read will block.

#### Exceptions

<b>IOException</b> (p. 2210)	if an I/O error occurs.
---------------------------------	-------------------------

Reimplemented from **decaf::io::Reader** (p. 3259).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**InputStreamReader.h**

## 6.403 decaf::internal::nio::IntArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/IntArrayBuffer.h>
```

Inheritance diagram for decaf::internal::nio::IntArrayBuffer:

#### Public Member Functions

- **IntArrayBuffer** (int size, bool readOnly=false) throw ( decaf::lang::exceptions::IllegalArgumentException )  
Creates a **IntArrayBuffer** (p. 2119) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.
- **IntArrayBuffer** (int \*array, int size, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
Creates a **IntArrayBuffer** (p. 2119) object that wraps the given array.
- **IntArrayBuffer** (const decaf::lang::Pointer< ByteArrayAdapter > &array, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

- **IntArrayBuffer** (const **IntArrayBuffer** &other)

Create a **IntArrayBuffer** (p. 2119) that mirrors this one, meaning it shares a reference to this buffers `ByteArrayAdapter` and when changes are made to that data it is reflected in both.

- virtual **~IntArrayBuffer** ()

- virtual int \* **array** () throw ( `decaf::lang::exceptions::UnsupportedOperationException`, `decaf::nio::ReadOnlyBufferException` )

Returns the int array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns**

the array that backs this **Buffer** (p. 936).

**Exceptions**

<b>ReadOnlyBufferException</b> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<b>UnsupportedOperationException</b>	if the underlying store has no array.

- virtual int **arrayOffset** () throw ( `decaf::lang::exceptions::UnsupportedOperationException`, `decaf::nio::ReadOnlyBufferException` )

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns**

The offset into the backing array where index zero starts.

**Exceptions**

<b>ReadOnlyBufferException</b> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<b>UnsupportedOperationException</b>	if the underlying store has no array.

- virtual **IntBuffer \* asReadOnlyBuffer** () const

Creates a new, read-only int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those

of this buffer.

#### Returns

The new, read-only int buffer which the caller then owns.

- virtual IntBuffer & **compact** () throw ( decaf::nio::ReadOnlyBufferException )

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 941) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 940) - 1 is copied to index  $n = \text{limit}()$  (p. 940) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

#### Returns

a reference to this **IntBuffer** (p. 2130)

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only.
---	------------------------------

- virtual IntBuffer \* **duplicate** ()

Creates a new int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

a new int **Buffer** (p. 936) which the caller owns.

- virtual int **get** () throw ( decaf::nio::BufferUnderflowException )

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

#### Returns

the int at the current position.

#### Exceptions

<b>BufferUnderflowException</b> (p. 967)	if there no more data to return.
---	----------------------------------

- virtual int **get** (int index) const throw ( lang::exceptions::IndexOutOfBoundsException )

Absolute get method.

Reads the value at the given index.

**Parameters**

index	The index in the <b>Buffer</b> (p. 936) where the int is to be read.
-------	--

**Returns**

the int that is located at the given index.

**Exceptions**

IndexOutOfBoundsException	if index is not smaller than the buffer's limit, or index is negative.
---------------------------	--

- virtual bool **hasArray** () const

Tells whether or not this buffer is backed by an accessible int array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns**

true if, and only if, this buffer is backed by an array and is not read-only.

- virtual bool **isReadOnly** () const

Tells whether or not this buffer is read-only.

**Returns**

true if, and only if, this buffer is read-only.

- virtual IntBuffer & **put** (int value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

Writes the given integer into this buffer at the current position, and then increments the position.

**Parameters**

value	The integer value to be written.
-------	----------------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b>BufferOverflowException</b> (p. 964)	if this buffer's current position is not smaller than its limit.
<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only.

- virtual IntBuffer & **put** (int index, int value) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

Writes the given ints into this buffer at the given index.

**Parameters**

index	The position in the <b>Buffer</b> (p. 936) to write the data.
value	The ints to write.

**Returns**

a reference to this buffer.

**Exceptions**

IndexOutOfBoundsException	- If index greater than the buffer's limit minus the size of the type being written, or the index is negative.
<b>ReadOnlyBufferException</b> (p. 3260)	- If this buffer is read-only.

- virtual IntBuffer \* **slice** () const

Creates a new **IntBuffer** (p. 2130) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns**

the newly create **IntBuffer** (p. 2130) which the caller owns.

**Protected Member Functions**

- virtual void **setReadOnly** (bool value)

Sets this **IntArrayBuffer** (p. 2119) as Read-Only.

**6.403.1 Constructor & Destructor Documentation**

**6.403.1.1** decaf::internal::nio::IntArrayBuffer::IntArrayBuffer ( int size, bool readOnly = false ) throw ( decaf::lang::exceptions::IllegalArgumentException )

Creates a **IntArrayBuffer** (p. 2119) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

**Parameters**

size	The size of the array, this is the limit we read and write to.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

**Exceptions**

IllegalArgumentEx- ception	if the capacity value is negative.
-------------------------------	------------------------------------

**6.403.1.2** `decaf::internal::nio::IntArrayBuffer::IntArrayBuffer ( int * array, int size, int offset, int length, bool readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a **IntArrayBuffer** (p. 2119) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

#### Parameters

<i>array</i>	The actual array to wrap.
<i>size</i>	The size of the given array.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if offset is greater than array capacity.

**6.403.1.3** `decaf::internal::nio::IntArrayBuffer::IntArrayBuffer ( const decaf::lang::Pointer< ByteArrayAdapter > & array, int offset, int length, bool readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **IntArrayBuffer** (p. 2119) will be that of the remaining capacity of the passed buffer.

#### Parameters

<i>array</i>	The ByteArrayAdapter to wrap.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>NullPointerException</i>	if array is NULL
<i>IndexOutOfBoundsException</i>	if offset + length is greater than array size.

6.403.1.4 `decaf::internal::nio::IntArrayBuffer::IntArrayBuffer ( const IntArrayBuffer & other )`

Create a **IntArrayBuffer** (p. 2119) that mirrors this one, meaning it shares a reference to this buffers `ByteArrayAdapter` and when changes are made to that data it is reflected in both.

#### Parameters

<i>other</i>	The <b>IntArrayBuffer</b> (p. 2119) this one is to mirror.
--------------	--

6.403.1.5 `virtual decaf::internal::nio::IntArrayBuffer::~~IntArrayBuffer ( ) [virtual]`

### 6.403.2 Member Function Documentation

6.403.2.1 `virtual int* decaf::internal::nio::IntArrayBuffer::array ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Returns the int array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

the array that backs this **Buffer** (p. 936).

#### Exceptions

<i>ReadOnlyBufferException</i> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implements **decaf::nio::IntBuffer** (p. 2134).

6.403.2.2 `virtual int decaf::internal::nio::IntArrayBuffer::arrayOffset ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns**

The offset into the backing array where index zero starts.

**Exceptions**

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<i><b>UnsupportedOperationException</b></i>	if the underlying store has no array.

Implements **decaf::nio::IntBuffer** (p. 2134).

**6.403.2.3** `virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::asReadOnlyBuffer ( ) const`  
[virtual]

Creates a new, read-only int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

**Returns**

The new, read-only int buffer which the caller then owns.

Implements **decaf::nio::IntBuffer** (p. 2135).

**6.403.2.4** `virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::compact ( ) throw ( decaf::nio::ReadOnlyBufferException )` [virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 941) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 940) - 1 is copied to index  $n = \text{limit}()$  (p. 940) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.



**Returns**

a reference to this **IntBuffer** (p. 2130)

**Exceptions**

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this buffer is read-only.
--	------------------------------

Implements **decaf::nio::IntBuffer** (p. 2135).

**6.403.2.5 virtual IntBuffer\* decaf::internal::nio::IntArrayBuffer::duplicate ( ) [virtual]**

Creates a new int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns**

a new int **Buffer** (p. 936) which the caller owns.

Implements **decaf::nio::IntBuffer** (p. 2136).

**6.403.2.6 virtual int decaf::internal::nio::IntArrayBuffer::get ( ) throw ( decaf::nio::BufferUnderflowException ) [virtual]**

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

**Returns**

the int at the current position.

**Exceptions**

<i><b>BufferUnderflowException</b></i> (p. 967)	if there no more data to return.
--	----------------------------------

Implements **decaf::nio::IntBuffer** (p. 2138).

**6.403.2.7** `virtual int decaf::internal::nio::IntArrayBuffer::get ( int index ) const throw ( lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Absolute get method.

Reads the value at the given index.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the int is to be read.
--------------	--

#### Returns

the int that is located at the given index.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implements **decaf::nio::IntBuffer** (p. 2136).

**6.403.2.8** `virtual bool decaf::internal::nio::IntArrayBuffer::hasArray ( ) const [inline, virtual]`

Tells whether or not this buffer is backed by an accessible int array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

#### Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implements **decaf::nio::IntBuffer** (p. 2138).

**6.403.2.9** `virtual bool decaf::internal::nio::IntArrayBuffer::isReadOnly ( ) const [inline, virtual]`

Tells whether or not this buffer is read-only.

#### Returns

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 940).

6.403.2.10 virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::put ( int *index*, int *value* ) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException ) [virtual]

Writes the given ints into this buffer at the given index.

#### Parameters

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data.
<i>value</i>	The ints to write.

#### Returns

a reference to this buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	- If index greater than the buffer's limit minus the size of the type being written, or the index is negative.
<i>ReadOnlyBufferException</i> (p. 3260)	- If this buffer is read-only.

Implements **decaf::nio::IntBuffer** (p. 2139).

6.403.2.11 virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::put ( int *value* ) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException ) [virtual]

Writes the given integer into this buffer at the current position, and then increments the position.

#### Parameters

<i>value</i>	The integer value to be written.
--------------	----------------------------------

#### Returns

a reference to this buffer.

#### Exceptions

<i>BufferOverflowException</i> (p. 964)	if this buffer's current position is not smaller than its limit.
<i>ReadOnlyBufferException</i> (p. 3260)	if this buffer is read-only.

Implements **decaf::nio::IntBuffer** (p. 2138).

**6.403.2.12** `virtual void decaf::internal::nio::IntArrayBuffer::setReadOnly ( bool value )`  
`[inline, protected, virtual]`

Sets this **IntArrayBuffer** (p. 2119) as Read-Only.

#### Parameters

<i>value</i>	Boolean value, true if this buffer is to be read-only, false otherwise.
--------------	---

**6.403.2.13** `virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::slice ( ) const`  
`[virtual]`

Creates a new **IntBuffer** (p.2130) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the newly create **IntBuffer** (p. 2130) which the caller owns.

Implements **decaf::nio::IntBuffer** (p. 2141).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/IntArrayBuffer.h`

## 6.404 decaf::nio::IntBuffer Class Reference

This class defines four categories of operations upon int buffers:

```
#include <src/main/decaf/nio/IntBuffer.h>
```

Inheritance diagram for `decaf::nio::IntBuffer`:

#### Public Member Functions

- `virtual ~IntBuffer ()`
- `virtual std::string toString () const`
- `virtual int * array ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )`

*Returns the int array that backs this buffer (optional operation).*

- virtual int **arrayOffset** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )

*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*

- virtual **IntBuffer** \* **asReadOnlyBuffer** () const =0

*Creates a new, read-only int buffer that shares this buffer's content.*

- virtual **IntBuffer** & **compact** ()=0 throw ( ReadOnlyBufferException )

*Compacts this buffer.*

- virtual **IntBuffer** \* **duplicate** ()=0

*Creates a new int buffer that shares this buffer's content.*

- virtual int **get** ()=0 throw ( BufferUnderflowException )

*Relative get method.*

- virtual int **get** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Absolute get method.*

- **IntBuffer** & **get** (std::vector< int > buffer) throw ( BufferUnderflowException )

*Relative bulk get method.*

- **IntBuffer** & **get** (int \*buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*Relative bulk get method.*

- virtual bool **hasArray** () const =0

*Tells whether or not this buffer is backed by an accessible int array.*

- **IntBuffer** & **put** (**IntBuffer** &src) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )

*This method transfers the ints remaining in the given source buffer into this buffer.*

- **IntBuffer** & **put** (const int \*buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*This method transfers ints into this buffer from the given source array.*

- **IntBuffer** & **put** (std::vector< int > &buffer) throw ( BufferOverflowException, ReadOnlyBufferException )

*This method transfers the entire content of the given source ints array into this buffer.*

- virtual **IntBuffer** & **put** (int value)=0 throw ( **BufferOverflowException**, **ReadOnlyBufferException** )

*Writes the given integer into this buffer at the current position, and then increments the position.*

- virtual **IntBuffer** & **put** (int index, int value)=0 throw ( **decaf::lang::exceptions::IndexOutOfBoundsException**, **ReadOnlyBufferException** )

*Writes the given ints into this buffer at the given index.*

- virtual **IntBuffer** \* **slice** () const =0

*Creates a new **IntBuffer** (p.2130) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **IntBuffer** &value) const

- virtual bool **equals** (const **IntBuffer** &value) const

- virtual bool **operator==** (const **IntBuffer** &value) const

- virtual bool **operator<** (const **IntBuffer** &value) const

### Static Public Member Functions

- static **IntBuffer** \* **allocate** (int capacity) throw ( **decaf::lang::exceptions::IllegalArgumentException** )

*Allocates a new Double buffer.*

- static **IntBuffer** \* **wrap** (int \*array, int size, int offset, int length) throw ( **decaf::lang::exceptions::NullPointerException**, **decaf::lang::exceptions::IndexOutOfBoundsException** )

*Wraps the passed buffer with a new **IntBuffer** (p.2130).*

- static **IntBuffer** \* **wrap** (std::vector< int > &buffer)

*Wraps the passed STL int Vector in a **IntBuffer** (p.2130).*

### Protected Member Functions

- **IntBuffer** (int capacity) throw ( **decaf::lang::exceptions::IllegalArgumentException** )

*Creates a **IntBuffer** (p. 2130) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

### 6.404.1 Detailed Description

This class defines four categories of operations upon int buffers: o Absolute and relative get and put methods that read and write single ints; o Relative bulk get methods that transfer contiguous sequences of ints from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of ints from a int array or some other int buffer into this buffer o Methods for compacting, duplicating, and slicing a int buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing int array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

### 6.404.2 Constructor & Destructor Documentation

**6.404.2.1** `decaf::nio::IntBuffer::IntBuffer ( int capacity ) throw ( decaf::lang::exceptions::IllegalArgumentException )` [protected]

Creates a **IntBuffer** (p. 2130) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

#### Parameters

<i>capacity</i>	The size and limit of the <b>Buffer</b> (p. 936) in integers.
-----------------	---

#### Exceptions

<i>IllegalArgumentEx-ception</i>	if capacity is negative.
----------------------------------	--------------------------

**6.404.2.2** `virtual decaf::nio::IntBuffer::~~IntBuffer ( )` [inline, virtual]

### 6.404.3 Member Function Documentation

**6.404.3.1** `static IntBuffer* decaf::nio::IntBuffer::allocate ( int capacity ) throw ( decaf::lang::exceptions::IllegalArgumentException )` [static]

Allocates a new Double buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will

be undefined. It will have a backing array, and its array offset will be zero.

#### Parameters

<i>capacity</i>	The size of the Double buffer in integers.
-----------------	--

#### Returns

the **IntBuffer** (p. 2130) that was allocated, caller owns.

**6.404.3.2** `virtual int* decaf::nio::IntBuffer::array ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the int array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

the array that backs this **Buffer** (p. 936).

#### Exceptions

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<i><b>UnsupportedOperationException</b></i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2125).

**6.404.3.3** `virtual int decaf::nio::IntBuffer::arrayOffset ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

The offset into the backing array where index zero starts.



**Exceptions**

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<i><b>UnsupportedOperationException</b></i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2125).

**6.404.3.4** `virtual IntBuffer* decaf::nio::IntBuffer::asReadOnlyBuffer ( ) const` [pure virtual]

Creates a new, read-only int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

**Returns**

The new, read-only int buffer which the caller then owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2126).

**6.404.3.5** `virtual IntBuffer& decaf::nio::IntBuffer::compact ( ) throw ( ReadOnlyBufferException )` [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 941) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 940) - 1 is copied to index  $n = \text{limit}()$  (p. 940) - 1 -  $p$ . The buffer's position is then set to  $n + 1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

**Returns**

a reference to this **IntBuffer** (p. 2130)

**Exceptions**

<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.
--	------------------------------

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2126).

**6.404.3.6** `virtual int decaf::nio::IntBuffer::compareTo ( const IntBuffer & value ) const`  
[virtual]

**6.404.3.7** `virtual IntBuffer* decaf::nio::IntBuffer::duplicate ( )` [pure virtual]

Creates a new int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns**

a new int **Buffer** (p. 936) which the caller owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2127).

**6.404.3.8** `virtual bool decaf::nio::IntBuffer::equals ( const IntBuffer & value ) const`  
[virtual]

**6.404.3.9** `IntBuffer& decaf::nio::IntBuffer::get ( std::vector< int > buffer ) throw ( BufferUnderflowException )`

Relative bulk get method.

This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize( N )` before calling this get method.

**Returns**

a reference to this **Buffer** (p. 936).

**Exceptions**

<b><i>BufferUnderflowException</i></b> (p. 967)	if there are fewer than length ints remaining in this buffer.
--	---

**6.404.3.10** `virtual int decaf::nio::IntBuffer::get ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [pure virtual]

Absolute get method.

Reads the value at the given index.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the int is to be read.
--------------	--

#### Returns

the int that is located at the given index.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2127).

**6.404.3.11** `IntBuffer& decaf::nio::IntBuffer::get ( int * buffer, int size, int offset, int length ) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )`

Relative bulk get method.

This method transfers ints from this buffer into the given destination array. If there are fewer ints remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 941), then no bytes are transferred and a **BufferUnderflowException** (p. 967) is thrown.

Otherwise, this method copies `length` ints from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

#### Parameters

<i>buffer</i>	The pointer to an allocated buffer to fill.
<i>size</i>	The size of the buffer that was passed in.
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The amount of data to put in the passed buffer.

#### Returns

a reference to this **Buffer** (p. 936).

#### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 967)	if there are fewer than length ints remaining in this buffer.
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

**6.404.3.12** `virtual int decaf::nio::IntBuffer::get ( ) throw ( BufferUnderflowException )`  
[pure virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

#### Returns

the int at the current position.

#### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 967)	if there no more data to return.
--	----------------------------------

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2127).

**6.404.3.13** `virtual bool decaf::nio::IntBuffer::hasArray ( ) const` [pure virtual]

Tells whether or not this buffer is backed by an accessible int array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

#### Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2128).

6.404.3.14 `virtual bool decaf::nio::IntBuffer::operator< ( const IntBuffer & value ) const`  
[virtual]

6.404.3.15 `virtual bool decaf::nio::IntBuffer::operator== ( const IntBuffer & value ) const`  
[virtual]

6.404.3.16 `virtual IntBuffer& decaf::nio::IntBuffer::put ( int value ) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes the given integer into this buffer at the current position, and then increments the position.

#### Parameters

<i>value</i>	The integer value to be written.
--------------	----------------------------------

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 964)	if this buffer's current position is not smaller than its limit.
<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2129).

6.404.3.17 `virtual IntBuffer& decaf::nio::IntBuffer::put ( int index, int value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )` [pure virtual]

Writes the given ints into this buffer at the given index.

#### Parameters

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data.
<i>value</i>	The ints to write.

#### Returns

a reference to this buffer.

#### Exceptions

<b>IndexOutOfBoundsException</b>	- If index greater than the buffer's limit minus the size of the type being written, or the index is negative.
----------------------------------	--

<b><i>ReadOnlyBufferException</i></b> (p. 3260)	- If this buffer is read-only.
--	--------------------------------

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2128).

**6.404.3.18** **IntBuffer& decaf::nio::IntBuffer::put ( const int \* *buffer*, int *size*, int *offset*, int *length* ) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )**

This method transfers ints into this buffer from the given source array.

If there are more ints to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 941), then no ints are transferred and a **BufferOverflowException** (p. 964) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

#### Parameters

<i>buffer</i>	The array from which integers are to be read.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The offset within the array of the first char to be read.
<i>length</i>	The number of integers to be read from the given array.

#### Returns

a reference to this buffer.

#### Exceptions

<b><i>BufferOverflowException</i></b> (p. 964)	if there is insufficient space in this buffer.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of <code>size</code> , <code>offset</code> , or <code>length</code> are not met.

### 6.404.3.19 IntBuffer& decaf::nio::IntBuffer::put ( std::vector< int > & *buffer* ) throw ( BufferOverflowException, ReadOnlyBufferException )

This method transfers the entire content of the given source ints array into this buffer.

This is the same as calling put( &buffer[0], 0, buffer.size()).

#### Parameters

<i>buffer</i>	The buffer whose contents are copied to this <b>IntBuffer</b> (p. 2130).
---------------	--

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 964)	if there is insufficient space in this buffer.
<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only.

### 6.404.3.20 IntBuffer& decaf::nio::IntBuffer::put ( IntBuffer & *src* ) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )

This method transfers the ints remaining in the given source buffer into this buffer.

If there are more ints remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 941), then no ints are transferred and a **BufferOverflowException** (p. 964) is thrown.

Otherwise, this method copies `n = src.remaining()` ints from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

#### Parameters

<i>src</i>	The buffer to take ints from an place in this one.
------------	--

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 964)	if there is insufficient space in this buffer for the remaining ints in the source buffer.
<b>IllegalArgumentException</b>	if the source buffer is this buffer.

<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.
--	------------------------------

**6.404.3.21** `virtual IntBuffer* decaf::nio::IntBuffer::slice ( ) const` [pure virtual]

Creates a new **IntBuffer** (p. 2130) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the newly create **IntBuffer** (p. 2130) which the caller owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2130).

**6.404.3.22** `virtual std::string decaf::nio::IntBuffer::toString ( ) const` [virtual]

#### Returns

a std::string describing this object

**6.404.3.23** `static IntBuffer* decaf::nio::IntBuffer::wrap ( int * array, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [static]

Wraps the passed buffer with a new **IntBuffer** (p. 2130).

The new buffer will be backed by the given int array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>array</i>	The array that will back the new buffer.
<i>size</i>	The size of the passed in array.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.



**Returns**

a new **IntBuffer** (p. 2130) that is backed by buffer, caller owns.

**Exceptions**

<i>NullPointerException</i>	if the array pointer is NULL.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

**6.404.3.24 static IntBuffer\* decaf::nio::IntBuffer::wrap ( std::vector< int > & buffer )**  
[static]

Wraps the passed STL int Vector in a **IntBuffer** (p. 2130).

The new buffer will be backed by the given int array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters**

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).
---------------	--

**Returns**

a new **IntBuffer** (p. 2130) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**IntBuffer.h**

**6.405 decaf::lang::Integer Class Reference**

```
#include <src/main/decaf/lang/Integer.h>
```

Inheritance diagram for decaf::lang::Integer:

**Public Member Functions**

- **Integer** (int value)
- **Integer** (const std::string &value) throw ( exceptions::NumberFormatException )
- virtual ~**Integer** ()

- virtual int **compareTo** (const **Integer** &i) const  
*Compares this **Integer** (p. 2143) instance with another.*
- bool **equals** (const **Integer** &i) const
- virtual bool **operator==** (const **Integer** &i) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Integer** &i) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual int **compareTo** (const int &i) const  
*Compares this **Integer** (p. 2143) instance with another.*
- bool **equals** (const int &i) const
- virtual bool **operator==** (const int &i) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const int &i) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- std::string **toString** () const
- virtual double **doubleValue** () const  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const  
*Answers the float value which the receiver represents.*
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual short **shortValue** () const  
*Answers the short value which the receiver represents.*
- virtual int **intValue** () const  
*Answers the int value which the receiver represents.*
- virtual long long **longValue** () const  
*Answers the long value which the receiver represents.*

### Static Public Member Functions

- static **Integer decode** (const std::string &value) throw ( exceptions::NumberFormatException )

Decodes a *String* (p. 3775) into a *Integer* (p. 2143).

- static int **reverseBytes** (int value)  
*Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified int value.*
- static int **reverse** (int value)  
*Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified int value.*
- static int **parseInt** (const std::string &s, int radix) throw ( exceptions::NumberFormatException )  
*Parses the string argument as a signed int in the radix specified by the second argument.*
- static int **parseInt** (const std::string &s) throw ( exceptions::NumberFormatException )  
*Parses the string argument as a signed decimal int.*
- static **Integer** **valueOf** (int value)  
*Returns a *Integer* (p. 2143) instance representing the specified int value.*
- static **Integer** **valueOf** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Returns a *Integer* (p. 2143) object holding the value given by the specified std::string.*
- static **Integer** **valueOf** (const std::string &value, int radix) throw ( exceptions::NumberFormatException )  
*Returns a *Integer* (p. 2143) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*
- static int **bitCount** (int value)  
*Returns the number of one-bits in the two's complement binary representation of the specified int value.*
- static std::string **toString** (int value)  
*Converts the int to a *String* (p. 3775) representation.*
- static std::string **toString** (int value, int radix)  
*Returns a string representation of the first argument in the radix specified by the second argument.*
- static std::string **toHexString** (int value)  
*Returns a string representation of the integer argument as an unsigned integer in base 16.*
- static std::string **toOctalString** (int value)

*Returns a string representation of the integer argument as an unsigned integer in base 8.*

- static std::string **toBinaryString** (int value)

*Returns a string representation of the integer argument as an unsigned integer in base 2.*

- static int **highestOneBit** (int value)

*Returns an int value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.*

- static int **lowestOneBit** (int value)

*Returns an int value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.*

- static int **numberOfLeadingZeros** (int value)

*Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value.*

- static int **numberOfTrailingZeros** (int value)

*Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value.*

- static int **rotateLeft** (int value, int distance)

*Returns the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits.*

- static int **rotateRight** (int value, int distance)

*Returns the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits.*

- static int **signum** (int value)

*Returns the signum function of the specified int value.*

### Static Public Attributes

- static const int **SIZE** = 32

*The size in bits of the primitive int type.*

- static const int **MAX\_VALUE** = (int)0x7FFFFFFF

*The maximum value that the primitive type can hold.*

- static const int **MIN\_VALUE** = (int)0x80000000

*The minimum value that the primitive type can hold.*

## 6.405.1 Constructor & Destructor Documentation

### 6.405.1.1 decaf::lang::Integer::Integer ( int *value* )

#### Parameters

<i>value</i>	The primitive value to wrap in an <b>Integer</b> (p. 2143) instance.
--------------	--

### 6.405.1.2 decaf::lang::Integer::Integer ( const std::string & *value* ) throw ( exceptions::NumberFormatException )

#### Parameters

<i>value</i>	The base 10 encoded string to decode to an <b>Integer</b> (p. 2143) and wrap.
--------------	---

#### Exceptions

<i>NumberFormatException</i>	
------------------------------	--

### 6.405.1.3 virtual decaf::lang::Integer::~~Integer ( ) [inline, virtual]

## 6.405.2 Member Function Documentation

### 6.405.2.1 static int decaf::lang::Integer::bitCount ( int *value* ) [static]

Returns the number of one-bits in the two's complement binary representation of the specified int value.

This function is sometimes referred to as the population count.

#### Parameters

<i>value</i>	- the int to count
--------------	--------------------

#### Returns

the number of one-bits in the two's complement binary representation of the specified int value.

### 6.405.2.2 virtual unsigned char decaf::lang::Integer::byteValue ( ) const [inline, virtual]

Answers the byte value which the receiver represents.

#### Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2919).

**6.405.2.3** `virtual int decaf::lang::Integer::compareTo ( const int & i ) const` [virtual]

Compares this **Integer** (p. 2143) instance with another.

#### Parameters

<i>i</i>	- the <b>Integer</b> (p. 2143) instance to be compared
----------	--

#### Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable< int >** (p. 1249).

**6.405.2.4** `virtual int decaf::lang::Integer::compareTo ( const Integer & i ) const`  
[virtual]

Compares this **Integer** (p. 2143) instance with another.

#### Parameters

<i>i</i>	- the <b>Integer</b> (p. 2143) instance to be compared
----------	--

#### Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable< Integer >** (p. 1249).

**6.405.2.5** `static Integer decaf::lang::Integer::decode ( const std::string & value ) throw ( exceptions::NumberFormatException )` [static]

Decodes a **String** (p. 3775) into a **Integer** (p. 2143).

Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the Integer.parseInt method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a NumberFormatException will be thrown. The result is negated if first character of the specified **String** (p. 3775) is the minus sign. No whitespace characters are permitted in the string.

**Parameters**

<i>value</i>	- The string to decode
--------------	------------------------

**Returns**

a **Integer** (p. 2143) object containing the decoded value

**Exceptions**

<i>NumberFormatException</i>	if the string is not formatted correctly.
------------------------------	---

**6.405.2.6** `virtual double decaf::lang::Integer::doubleValue ( ) const` `[inline, virtual]`

Answers the double value which the receiver represents.

**Returns**

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2919).

**6.405.2.7** `bool decaf::lang::Integer::equals ( const Integer & i ) const` `[inline, virtual]`

**Parameters**

<i>i</i>	- the <b>Integer</b> (p. 2143) object to compare against.
----------	---

**Returns**

true if the two **Integer** (p. 2143) Objects have the same value.

Implements **decaf::lang::Comparable**< **Integer** > (p. 1250).

**6.405.2.8** `bool decaf::lang::Integer::equals ( const int & i ) const` `[inline, virtual]`

**Parameters**

<i>i</i>	- the <b>Integer</b> (p. 2143) object to compare against.
----------	---

**Returns**

true if the two **Integer** (p. 2143) Objects have the same value.

Implements **decaf::lang::Comparable**< **int** > (p. 1250).

**6.405.2.9** `virtual float decaf::lang::Integer::floatValue ( ) const` `[inline, virtual]`

Answers the float value which the receiver represents.

#### Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2920).

**6.405.2.10** `static int decaf::lang::Integer::highestOneBit ( int value )` `[static]`

Returns an int value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.

Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

#### Parameters

<i>value</i>	- the int to be inspected
--------------	---------------------------

#### Returns

an int value with a single one-bit, in the position of the highest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

**6.405.2.11** `virtual int decaf::lang::Integer::intValue ( ) const` `[inline, virtual]`

Answers the int value which the receiver represents.

#### Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2920).

**6.405.2.12** `virtual long long decaf::lang::Integer::longValue ( ) const` `[inline, virtual]`

Answers the long value which the receiver represents.

#### Returns

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2920).



**6.405.2.13 static int decaf::lang::Integer::lowestOneBit ( int *value* ) [static]**

Returns an int value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.

Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

**Parameters**

<i>value</i>	- the int to be inspected
--------------	---------------------------

**Returns**

an int value with a single one-bit, in the position of the lowest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

**6.405.2.14 static int decaf::lang::Integer::numberOfLeadingZeros ( int *value* ) [static]**

Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value.

Returns 32 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Note that this method is closely related to the logarithm base 2. For all positive int values  $x$ :

$\ast \text{floor}(\log_2(x)) = 31 - \text{numberOfLeadingZeros}(x)$   $\ast \text{ceil}(\log_2(x)) = 32 - \text{numberOfLeadingZeros}(x - 1)$

**Parameters**

<i>value</i>	- the int to be inspected
--------------	---------------------------

**Returns**

the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value, or 32 if the value is equal to zero.

**6.405.2.15 static int decaf::lang::Integer::numberOfTrailingZeros ( int *value* ) [static]**

Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value.

Returns 32 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

**Parameters**

<i>value</i>	- the int to be inspected
--------------	---------------------------

**Returns**

the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value, or 32 if the value is equal to zero.

**6.405.2.16** `virtual bool decaf::lang::Integer::operator< ( const int & i ) const` `[inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

**Parameters**

<i>i</i>	- the value to be compared to this one.
----------	---

**Returns**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< int >** (p. 1250).

**6.405.2.17** `virtual bool decaf::lang::Integer::operator< ( const Integer & i ) const` `[inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

**Parameters**

<i>i</i>	- the value to be compared to this one.
----------	---

**Returns**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Integer >** (p. 1250).

**6.405.2.18** `virtual bool decaf::lang::Integer::operator== ( const Integer & i ) const` `[inline, virtual]`

Compares equality between this object and the one passed.

**Parameters**

<i>i</i>	- the value to be compared to this one.
----------	---

**Returns**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Integer** > (p. 1250).

**6.405.2.19** `virtual bool decaf::lang::Integer::operator==( const int & i ) const [inline, virtual]`

Compares equality between this object and the one passed.

**Parameters**

<i>i</i>	- the value to be compared to this one.
----------	---

**Returns**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **int** > (p. 1250).

**6.405.2.20** `static int decaf::lang::Integer::parseInt ( const std::string & s, int radix ) throw ( exceptions::NumberFormatException ) [static]`

Parses the string argument as a signed int in the radix specified by the second argument.

The characters in the string must all be digits, of the specified radix (as determined by whether **Character.digit(char, int)** (p. 1130) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type **NumberFormatException** is thrown if any of the following situations occurs: \* The first argument is null or is a string of length zero. \* The radix is either smaller than **Character.MIN\_RADIX** (p. 1134) or larger than **Character.MAX\_RADIX** (p. 1134). \* Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. \* The value represented by the string is not a value of type int.

**Parameters**

<i>s</i>	- the <b>String</b> (p. 3775) containing the int representation to be parsed
<i>radix</i>	- the radix to be used while parsing s

**Returns**

the int represented by the string argument in the specified radix.

**Exceptions**

<i>NumberFormatException</i>	- If <b>String</b> (p. 3775) does not contain a parsable int.
------------------------------	---

**6.405.2.21** `static int decaf::lang::Integer::parseInt ( const std::string & s ) throw ( exceptions::NumberFormatException )` [static]

Parses the string argument as a signed decimal int.

The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting int value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseInt( const std::string, int )` method.

#### Parameters

<i>s</i>	- <b>String</b> (p. 3775) to convert to a int
----------	---

#### Returns

the converted int value

#### Exceptions

<i>NumberFormatException</i>	if the string is not a int.
------------------------------	-----------------------------

**6.405.2.22** `static int decaf::lang::Integer::reverse ( int value )` [static]

Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified int value.

#### Parameters

<i>value</i>	- the value whose bits are to be reversed
--------------	---

#### Returns

the reversed bits int.

**6.405.2.23** `static int decaf::lang::Integer::reverseBytes ( int value )` [static]

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified int value.

#### Parameters

<i>value</i>	- the int whose bytes we are to reverse
--------------	---

#### Returns

the reversed int.

**6.405.2.24 static int decaf::lang::Integer::rotateLeft ( int *value*, int *distance* ) [static]**

Returns the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits.

(Bits shifted out of the left hand, or high-order, side reenter on the right, or low-order.)

Note that left rotation with a negative distance is equivalent to right rotation: rotateLeft(val, -distance) == rotateRight(val, distance). Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: rotateLeft(val, distance) == rotateLeft(val, distance & 0x1F).

**Parameters**

<i>value</i>	- the int to be inspected
<i>distance</i>	- the number of bits to rotate

**Returns**

the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits.

**6.405.2.25 static int decaf::lang::Integer::rotateRight ( int *value*, int *distance* ) [static]**

Returns the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits.

(Bits shifted out of the right hand, or low-order, side reenter on the left, or high-order.)

Note that right rotation with a negative distance is equivalent to left rotation: rotateRight(val, -distance) == rotateLeft(val, distance). Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: rotateRight(val, distance) == rotateRight(val, distance & 0x1F).

**Parameters**

<i>value</i>	- the int to be inspected
<i>distance</i>	- the number of bits to rotate

**Returns**

the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits.

**6.405.2.26 virtual short decaf::lang::Integer::shortValue ( ) const [inline, virtual]**

Answers the short value which the receiver represents.

**Returns**

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2920).

**6.405.2.27 static int decaf::lang::Integer::signum ( int *value* ) [static]**

Returns the signum function of the specified int value.

(The return value is -1 if the specified value is negative; 0 if the specified value is zero; and 1 if the specified value is positive.)

**Parameters**

<i>value</i>	- the int to be inspected
--------------	---------------------------

**Returns**

the signum function of the specified int value.

**6.405.2.28 static std::string decaf::lang::Integer::toBinaryString ( int *value* ) [static]**

Returns a string representation of the integer argument as an unsigned integer in base 2.

The unsigned integer value is the argument plus  $2^{32}$  if the argument is negative; otherwise it is equal to the argument. This value is converted to a string of ASCII digits in binary (base 2) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character.

The characters '0' and '1' are used as binary digits.

**Parameters**

<i>value</i>	- the int to be translated to a binary string
--------------	---

**Returns**

the unsigned int value as a binary string

**6.405.2.29 static std::string decaf::lang::Integer::toHexString ( int *value* ) [static]**

Returns a string representation of the integer argument as an unsigned integer in base 16.

The unsigned integer value is the argument plus  $2^{32}$  if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in hexadecimal (base 16) with no extra leading 0s. If the unsigned magnitude is zero,

it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as hexadecimal digits:

0123456789abcdef

If uppercase letters are desired, the toUpperCase() method may be called on the result:

#### Parameters

<i>value</i>	- the int to be translated to an Octal string
--------------	---

#### Returns

the unsigned int value as a Octal string

#### 6.405.2.30 static std::string decaf::lang::Integer::toOctalString ( int *value* ) [static]

Returns a string representation of the integer argument as an unsigned integer in base 8.

The unsigned integer value is the argument plus  $2^{32}$  if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in octal (base 8) with no extra leading 0s.

If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as octal digits:

01234567

#### Parameters

<i>value</i>	- the int to be translated to an Octal string
--------------	---

#### Returns

the unsigned int value as a Octal string

#### 6.405.2.31 std::string decaf::lang::Integer::toString ( ) const

#### Returns

this **Integer** (p. 2143) Object as a **String** (p. 3775) Representation

#### 6.405.2.32 static std::string decaf::lang::Integer::toString ( int *value*, int *radix* ) [static]

Returns a string representation of the first argument in the radix specified by the second argument.

If the radix is smaller than **Character.MIN\_RADIX** (p. 1134) or larger than **Character.MAX\_RADIX** (p. 1134), then the radix 10 is used instead.

If the first argument is negative, the first element of the result is the ASCII minus character '-'. If the first argument is not negative, no sign character appears in the result.

The remaining characters of the result represent the magnitude of the first argument. If the magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the magnitude will not be the zero character. The following ASCII characters are used as digits:

0123456789abcdefghijklmnopqrstuvwxyz

#### Parameters

<i>value</i>	- the int to convert to a string
<i>radix</i>	- the radix to format the string in

#### Returns

an int formatted to the string value of the radix given.

**6.405.2.33** `static std::string decaf::lang::Integer::toString ( int value ) [static]`

Converts the int to a **String** (p. 3775) representation.

#### Parameters

<i>value</i>	The int to convert to a <code>std::string</code> instance.
--------------	--

#### Returns

string representation

**6.405.2.34** `static Integer decaf::lang::Integer::valueOf ( const std::string & value, int radix ) throw ( exceptions::NumberFormatException ) [static]`

Returns a **Integer** (p. 2143) object holding the value extracted from the specified `std::string` when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed int in the radix specified by the second argument, exactly as if the argument were given to the `parseInt( std::string, int )` method. The result is a **Integer** (p. 2143) object that represents the int value specified by the string.

#### Parameters

<i>value</i>	- <code>std::string</code> to parse as base ( <i>radix</i> )
<i>radix</i>	- base of the string to parse.

#### Returns

new **Integer** (p. 2143) Object wrapping the primitive



**Exceptions**

<i>NumberFormatException</i>	if the string is not a valid int.
------------------------------	-----------------------------------

**6.405.2.35** `static Integer decaf::lang::Integer::valueOf ( int value ) [inline, static]`

Returns a **Integer** (p. 2143) instance representing the specified int value.

**Parameters**

<i>value</i>	- the int to wrap
--------------	-------------------

**Returns**

the new **Integer** (p. 2143) object wrapping value.

**6.405.2.36** `static Integer decaf::lang::Integer::valueOf ( const std::string & value ) throw ( exceptions::NumberFormatException ) [static]`

Returns a **Integer** (p. 2143) object holding the value given by the specified std::string.

The argument is interpreted as representing a signed decimal int, exactly as if the argument were given to the `parseInt( std::string )` method. The result is a **Integer** (p. 2143) object that represents the int value specified by the string.

**Parameters**

<i>value</i>	- std::string to parse as base 10
--------------	-----------------------------------

**Returns**

new **Integer** (p. 2143) Object wrapping the primitive

**Exceptions**

<i>NumberFormatException</i>	if the string is not a decimal int.
------------------------------	-------------------------------------

**6.405.3 Field Documentation**

**6.405.3.1** `const int decaf::lang::Integer::MAX_VALUE = (int)0xFFFFFFFF [static]`

The maximum value that the primitive type can hold.

**6.405.3.2** `const int decaf::lang::Integer::MIN_VALUE = (int)0x80000000` `[static]`

The minimum value that the primitive type can hold.

**6.405.3.3** `const int decaf::lang::Integer::SIZE = 32` `[static]`

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Integer.h`

## 6.406 `activemq::commands::IntegerResponse` Class Reference

```
#include <src/main/activemq/commands/IntegerResponse.h>
```

Inheritance diagram for `activemq::commands::IntegerResponse`:

### Public Member Functions

- `IntegerResponse ()`
- virtual `~IntegerResponse ()`
- virtual unsigned char `getDataStructureType ()` const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual `IntegerResponse * cloneDataStructure ()` const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void `copyDataStructure (const DataStructure *src)`  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string `toString ()` const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool `equals (const DataStructure *value)` const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual int `getResult ()` const
- virtual void `setResult (int result)`

### Static Public Attributes

- static const unsigned char **ID\_INTEGERRESPONSE** = 34

### Protected Attributes

- int **result**

### 6.406.1 Constructor & Destructor Documentation

6.406.1.1 **activemq::commands::IntegerResponse::IntegerResponse ( )**

6.406.1.2 **virtual activemq::commands::IntegerResponse::~~IntegerResponse ( )**  
[virtual]

### 6.406.2 Member Function Documentation

6.406.2.1 **virtual IntegerResponse\* activemq::commands::IntegerResponse::cloneDataStructure ( ) const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Reimplemented from **activemq::commands::Response** (p. 3380).

6.406.2.2 **virtual void activemq::commands::IntegerResponse::copyDataStructure ( const DataStructure \* *src* )** [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::Response** (p. 3380).

6.406.2.3 **virtual bool activemq::commands::IntegerResponse::equals ( const DataStructure \* *value* ) const** [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if `DataSet`'s are Equal.

Reimplemented from `activemq::commands::Response` (p. 3381).

**6.406.2.4** `virtual unsigned char activemq::commands::IntegerResponse::getDataStructureType ( ) const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

### Returns

new `DataSet` (p. 1713) type copy.

Reimplemented from `activemq::commands::Response` (p. 3381).

**6.406.2.5** `virtual int activemq::commands::IntegerResponse::getResult ( ) const` [virtual]

**6.406.2.6** `virtual void activemq::commands::IntegerResponse::setResult ( int result )` [virtual]

**6.406.2.7** `virtual std::string activemq::commands::IntegerResponse::toString ( ) const` [virtual]

Returns a string containing the information for this `DataSet` (p. 1713) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::Response` (p. 3382).

## 6.406.3 Field Documentation

**6.406.3.1** `const unsigned char activemq::commands::IntegerResponse::ID_INTEGERRESPONSE = 34` [static]

**6.406.3.2** `int activemq::commands::IntegerResponse::result` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/IntegerResponse.h`

6.407

activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller

Class Reference

6.407 — activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller <sup>2165</sup>

## Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2162).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/IntegerResponseMarshaller>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller:

## Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.407.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2162).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.407.2 Constructor & Destructor Documentation

6.407.2.1 **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::IntegerResponseMarshaller**  
 ( ) [inline]

6.407.2.2 **virtual activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::~~IntegerResponseMarshaller**  
 ( ) [inline, virtual]

### 6.407.3 Member Function Documentation

6.407.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller**  
 (p. 3415).

6.407.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller**  
 (p. 3415).

6.407.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::looseMarshal**  
 ( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

6.407

**activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller****Class Reference****2167****Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3415).

**6.407.3.4** virtual void **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3416).

**6.407.3.5** virtual int **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::tightMarshal1** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3417).

```
6.407.3.6  virtual void activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3417).

```
6.407.3.7  virtual void activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.



6.408

**activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller**

**Class Reference**

2169

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3418).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**IntegerResponseMarshaller.h**

## 6.408 **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2167).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/IntegerResponseMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller**:

### Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.408.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.2167).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.408.2 Constructor & Destructor Documentation

6.408.2.1 **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::IntegerResponseMarshaller**  
 ( ) [inline]

6.408.2.2 **virtual activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::~~IntegerResponseMarshaller**  
 ( ) [inline, virtual]

### 6.408.3 Member Function Documentation

6.408.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3395).

6.408.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

6.408

activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller

Class Reference

2171

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3395).

6.408.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3395).

6.408.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3396).

```

6.408.3.5  virtual int activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3397).

```

6.408.3.6  virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3397).

6.409

activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller

Class Reference

2173

6.408.3.7 virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3398).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**IntegerResponseMarshaller.h**

## 6.409 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2171).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/IntegerResponseMarshaller>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller**:

#### Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.409.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.2171).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.409.2 Constructor & Destructor Documentation

6.409.2.1 **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::IntegerResponseMarshaller**  
( ) [inline]

6.409.2.2 **virtual activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::~~IntegerResponseMarshaller**  
( ) [inline, virtual]

### 6.409.3 Member Function Documentation

6.409.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

6.409

**activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller**

**Class Reference**

2175

**Returns**

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3405).

**6.409.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3405).

**6.409.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3405).

**6.409.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3406).

```
6.409.3.5  virtual int activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3407).

```
6.409.3.6  virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---



## 6.410

**activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller**

**Class Reference**

**2177**

<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3407).

**6.409.3.7** virtual void **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3408).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**IntegerResponseMarshaller.h**

## 6.410 **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2175).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller**:

## Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.410.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.2175).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.410

activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller

Class Reference

2179

### 6.410.2 Constructor & Destructor Documentation

6.410.2.1 `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::IntegerResponseMarshaller ( ) [inline]`

6.410.2.2 `virtual activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::~~IntegerResponseMarshaller ( ) [inline, virtual]`

### 6.410.3 Member Function Documentation

6.410.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3390).

6.410.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3390).

6.410.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3390).

**6.410.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3391).

**6.410.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.410

**activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller**

**Class Reference**

**2181**

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3392).

**6.410.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3392).

**6.410.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3393).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h`

## 6.411 `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2180).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/IntegerResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller`:

### Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

## 6.411

### activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller Class Reference

2183

---

*Write a object instance to data output stream.*

#### 6.411.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2180).  
NOTE!: This file is auto generated - do not modify! if you need to make a change,  
please see the Java Classes in the activemq-openwire-generator module

#### 6.411.2 Constructor & Destructor Documentation

6.411.2.1 **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::IntegerResponseMarshaller**  
( ) [inline]

6.411.2.2 **virtual activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::~~IntegerResponseMarshaller**  
( ) [inline, virtual]

#### 6.411.3 Member Function Documentation

6.411.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

##### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller**  
(p. 3410).

6.411.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

##### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller**  
(p. 3410).

**6.411.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3410).

**6.411.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3411).

**6.411.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.



6.411

**activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller****Class Reference****2185****Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3412).

```
6.411.3.6 virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3412).

```
6.411.3.7 virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3413).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**IntegerResponseMarshaller.h**

## 6.412 **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller** Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2184).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/IntegerResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller**:

### Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )

## 6.412

### activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller

#### Class Reference

2187

---

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

#### 6.412.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2184).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

#### 6.412.2 Constructor & Destructor Documentation

6.412.2.1 **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::IntegerResponseMarshaller**  
( ) [**inline**]

6.412.2.2 **virtual activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::~~IntegerResponseMarshaller**  
( ) [**inline**, **virtual**]

#### 6.412.3 Member Function Documentation

6.412.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::createObject**  
( ) **const** [**virtual**]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3400).

**6.412.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3400).

**6.412.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3400).

**6.412.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## 6.412

**activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller**

**Class Reference**

**2189**

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3401).

**6.412.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3402).

**6.412.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3402).

```
6.412.3.7  virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
  * bs ) throw ( decaf::io::IOException )  [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3403).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/IntegerResponseMarshaller.h`

**6.413 internal\_state Struct Reference**

```
#include <src/main/decaf/internal/util/zip/deflate.h>
```

**Data Fields**

- `z_stream` strm
- `int` status
- `Bytef *` pending\_buf
- `ulg` pending\_buf\_size
- `Bytef *` pending\_out
- `uInt` pending
- `int` wrap
- `gz_headerp` gzhead

- **uInt** gzindex
- **Byte** method
- **int** last\_flush
- **uInt** w\_size
- **uInt** w\_bits
- **uInt** w\_mask
- **Bytef** \* window
- **ulg** window\_size
- **Posf** \* prev
- **Posf** \* head
- **uInt** ins\_h
- **uInt** hash\_size
- **uInt** hash\_bits
- **uInt** hash\_mask
- **uInt** hash\_shift
- **long** block\_start
- **uInt** match\_length
- **IPos** prev\_match
- **int** match\_available
- **uInt** strstart
- **uInt** match\_start
- **uInt** lookahead
- **uInt** prev\_length
- **uInt** max\_chain\_length
- **uInt** max\_lazy\_match
- **int** level
- **int** strategy
- **uInt** good\_match
- **int** nice\_match
- **struct** **ct\_data\_s** dyn\_ltree [HEAP\_SIZE]
- **struct** **ct\_data\_s** dyn\_dtree [2 \*D\_CODES+1]
- **struct** **ct\_data\_s** bl\_tree [2 \*BL\_CODES+1]
- **struct** **tree\_desc\_s** l\_desc
- **struct** **tree\_desc\_s** d\_desc
- **struct** **tree\_desc\_s** bl\_desc
- **ush** bl\_count [MAX\_BITS+1]
- **int** heap [2 \*L\_CODES+1]
- **int** heap\_len
- **int** heap\_max
- **uch** depth [2 \*L\_CODES+1]
- **uchf** \* l\_buf
- **uInt** lit\_bufsize
- **uInt** last\_lit
- **ushf** \* d\_buf
- **ulg** opt\_len
- **ulg** static\_len

- **uInt matches**
- **int last\_eob\_len**
- **ush bi\_buf**
- **int bi\_valid**
- **ulg high\_water**
- **int dummy**



### 6.413.1 Field Documentation

- 6.413.1.1 `ush internal_state::bi_buf`
- 6.413.1.2 `int internal_state::bi_valid`
- 6.413.1.3 `ush internal_state::bl_count[MAX_BITS+1]`
- 6.413.1.4 `struct tree_desc_s internal_state::bl_desc`
- 6.413.1.5 `struct ct_data_s internal_state::bl_tree[2 * BL_CODES+1]`
- 6.413.1.6 `long internal_state::block_start`
- 6.413.1.7 `ushf* internal_state::d_buf`
- 6.413.1.8 `struct tree_desc_s internal_state::d_desc`
- 6.413.1.9 `uch internal_state::depth[2 * L_CODES+1]`
- 6.413.1.10 `int internal_state::dummy`
- 6.413.1.11 `struct ct_data_s internal_state::dyn_dtree[2 * D_CODES+1]`
- 6.413.1.12 `struct ct_data_s internal_state::dyn_ltree[HEAP_SIZE]`
- 6.413.1.13 `uInt internal_state::good_match`
- 6.413.1.14 `gz_headerp internal_state::gzhead`
- 6.413.1.15 `uInt internal_state::gzindex`
- 6.413.1.16 `uInt internal_state::hash_bits`
- 6.413.1.17 `uInt internal_state::hash_mask`
- 6.413.1.18 `uInt internal_state::hash_shift`
- 6.413.1.19 `uInt internal_state::hash_size`
- 6.413.1.20 `Posf* internal_state::head`
- 6.413.1.21 `int internal_state::heap[2 * L_CODES+1]`
- 6.413.1.22 `int internal_state::heap_len`
- 6.413.1.23 `int internal_state::heap_max`
- 6.413.1.24 `ulg internal_state::high_water`

---

6.413.1.25 `uInt internal_state::ins_h`  
Generated on Tue Mar 5 2013 17:10:05 for adobe-emb-cpp-3.2.5 by Doxygen

- 6.413.1.26 `uchf* internal_state::l_buf`
- 6.413.1.27 `struct tree_desc_s internal_state::l_desc`
- 6.413.1.28 `int internal_state::last_eob_len`
- 6.413.1.29 `int internal_state::last_flush`

- src/main/decaf/internal/util/zip/**deflate.h**
- src/main/decaf/internal/util/zip/**zlib.h**

## 6.414 activemq::transport::mock::InternalCommandListener Class Reference

Listens for Commands sent from the **MockTransport** (p. 2854).

```
#include <src/main/activemq/transport/mock/InternalCommandListener.h>
```

Inheritance diagram for activemq::transport::mock::InternalCommandListener:

### Public Member Functions

- **InternalCommandListener** ()
- virtual **~InternalCommandListener** ()
- void **setTransport** (**MockTransport** \*transport)
- void **setResponseBuilder** (const **Pointer**< **ResponseBuilder** > &responseBuilder)
- virtual void **onCommand** (const **Pointer**< **Command** > &command)

*Event handler for the receipt of a command.*

- void **run** ()

*Default implementation of the run method - does nothing.*

### 6.414.1 Detailed Description

Listens for Commands sent from the **MockTransport** (p. 2854). This class processes all outbound commands and sends responses that are constructed by calling the Protocol provided **ResponseBuilder** (p. 3382) and getting a set of Commands to send back into the **MockTransport** (p. 2854) as incoming Commands and Responses.

### 6.414.2 Constructor & Destructor Documentation

6.414.2.1 `activemq::transport::mock::InternalCommandListener::InternalCommandListener ( )`

6.414.2.2 `virtual activemq::transport::mock::InternalCommandListener::~~InternalCommandListener ( ) [virtual]`

### 6.414.3 Member Function Documentation

6.414.3.1 `virtual void activemq::transport::mock::InternalCommandListener::onCommand ( const Pointer< Command > & command ) [virtual]`

Event handler for the receipt of a command.

The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3996) deletes the command upon receipt.

#### Parameters

<i>command</i>	the received command object.
----------------	------------------------------

Implements `activemq::transport::TransportListener` (p. 4014).

6.414.3.2 `void activemq::transport::mock::InternalCommandListener::run ( ) [virtual]`

Default implementation of the run method - does nothing.

Reimplemented from `decaf::lang::Thread` (p. 3884).

6.414.3.3 `void activemq::transport::mock::InternalCommandListener::setResponseBuilder ( const Pointer< ResponseBuilder > & responseBuilder ) [inline]`

6.414.3.4 `void activemq::transport::mock::InternalCommandListener::setTransport ( MockTransport * transport ) [inline]`

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/mock/InternalCommandListener.h`

## 6.415 decaf::lang::exceptions::InterruptedException Class Reference

```
#include <src/main/decaf/lang/exceptions/InterruptedException.h>
```

Inheritance diagram for `decaf::lang::exceptions::InterruptedException`:

## Public Member Functions

- **InterruptedException** () throw ()  
*Default Constructor.*
- **InterruptedException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1886).*
- **InterruptedException** (const **InterruptedException** &ex) throw ()  
*Copy Constructor.*
- **InterruptedException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **InterruptedException** (const std::exception \*cause) throw ()  
*Constructor.*
- **InterruptedException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **InterruptedException** \* **clone** () const  
*Clones this exception.*
- virtual ~**InterruptedException** () throw ()

### 6.415.1 Constructor & Destructor Documentation

6.415.1.1 `decaf::lang::exceptions::InterruptedException::InterruptedException ( ) throw ()`  
[inline]

Default Constructor.

6.415.1.2 `decaf::lang::exceptions::InterruptedException::InterruptedException ( const Exception & ex ) throw ()` [inline]

Conversion Constructor from some other **Exception** (p. 1886).

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

6.415.1.3 decaf::lang::exceptions::InterruptedException::InterruptedException ( const  
InterruptedException & ex ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

6.415.1.4 decaf::lang::exceptions::InterruptedException::InterruptedException ( const char \*  
*file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ... )  
throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.415.1.5 decaf::lang::exceptions::InterruptedException::InterruptedException ( const  
std::exception \* *cause* ) throw () [inline]

Constructor.

#### Parameters

<i>cause</i>	<b>Pointer</b> (p. 3032) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.415.1.6 decaf::lang::exceptions::InterruptedException::InterruptedException ( const char \*  
*file*, const int *lineNumber*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.415.1.7 `virtual decaf::lang::exceptions::InterruptedException::~~InterruptedException ( )  
throw () [inline, virtual]`

## 6.415.2 Member Function Documentation

6.415.2.1 `virtual InterruptedException* decaf::lang::exceptions::InterruptedException::clone ( ) const  
[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

an new **Exception** (p. 1886) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1889).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/InterruptedException.h`

## 6.416 decaf::io::InterruptedException Class Reference

```
#include <src/main/decaf/io/InterruptedException.h>
```

Inheritance diagram for `decaf::io::InterruptedException`:

### Public Member Functions

- **InterruptedException () throw ()**  
*Default Constructor.*
- **InterruptedException (const lang::Exception &ex) throw ()**  
*Copy Constructor.*
- **InterruptedException (const InterruptedException &ex) throw ()**  
*Copy Constructor.*
- **InterruptedException (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()**  
*Constructor - Initializes the file name and line number where this message occurred.*
- **InterruptedException (const std::exception \*cause) throw ()**

*Constructor.*

- **InterruptedException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()

*Constructor.*

- virtual **InterruptedException \* clone** () const

*Clones this exception.*

- virtual ~**InterruptedException** () throw ()

### 6.416.1 Constructor & Destructor Documentation

6.416.1.1 decaf::io::InterruptedException::InterruptedException ( ) throw () [inline]

Default Constructor.

6.416.1.2 decaf::io::InterruptedException::InterruptedException ( const lang::Exception & ex ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

6.416.1.3 decaf::io::InterruptedException::InterruptedException ( const InterruptedException & ex ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

6.416.1.4 decaf::io::InterruptedException::InterruptedException ( const char \* file, const int lineNumber, const std::exception \* cause, const char \* msg, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
-------------	--------------------------------------

<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.416.1.5** `decaf::io::InterruptedIOException::InterruptedIOException ( const std::exception *  
cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.416.1.6** `decaf::io::InterruptedIOException::InterruptedIOException ( const char * file, const  
int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor.

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.416.1.7** `virtual decaf::io::InterruptedIOException::~~InterruptedIOException ( ) throw ()  
[inline, virtual]`

## 6.416.2 Member Function Documentation

**6.416.2.1** `virtual InterruptedIOException* decaf::io::InterruptedIOException::clone ( )  
const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

a new exception that is a copy of this one.

Reimplemented from **decaf::io::IOException** (p. 2212).



Reimplemented in **decaf::net::SocketTimeoutException** (p. 3652).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**InterruptedIOException.h**

## 6.417 cms::InvalidClientIdException Class Reference

This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

```
#include <src/main/cms/InvalidClientIdException.h>
```

Inheritance diagram for cms::InvalidClientIdException:

### Public Member Functions

- **InvalidClientIdException** () throw ()
- **InvalidClientIdException** (const **InvalidClientIdException** &ex) throw ()
- **InvalidClientIdException** (const std::string &message, const std::exception \*cause) throw ()
- **InvalidClientIdException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**InvalidClientIdException** () throw ()

### 6.417.1 Detailed Description

This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

Since

1.3

## 6.417.2 Constructor & Destructor Documentation

- 6.417.2.1 `cms::InvalidClientIdException::InvalidClientIdException ( ) throw ()`
- 6.417.2.2 `cms::InvalidClientIdException::InvalidClientIdException ( const InvalidClientIdException & ex ) throw ()`
- 6.417.2.3 `cms::InvalidClientIdException::InvalidClientIdException ( const std::string & message, const std::exception * cause ) throw ()`
- 6.417.2.4 `cms::InvalidClientIdException::InvalidClientIdException ( const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace ) throw ()`
- 6.417.2.5 `virtual cms::InvalidClientIdException::~InvalidClientIdException ( ) throw ()`  
[virtual]

The documentation for this class was generated from the following file:

- `src/main/cms/InvalidClientIdException.h`

## 6.418 cms::InvalidDestinationException Class Reference

This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

```
#include <src/main/cms/InvalidDestinationException.h>
```

Inheritance diagram for cms::InvalidDestinationException:

### Public Member Functions

- **InvalidDestinationException** ( ) throw ()
- **InvalidDestinationException** (const **InvalidDestinationException** &ex) throw ()
- **InvalidDestinationException** (const std::string &message, const std::exception \*cause) throw ()
- **InvalidDestinationException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual **~InvalidDestinationException** ( ) throw ()

### 6.418.1 Detailed Description

This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

Since

1.3

## 6.418.2 Constructor & Destructor Documentation

6.418.2.1 `cms::InvalidDestinationException::InvalidDestinationException ( ) throw ()`

6.418.2.2 `cms::InvalidDestinationException::InvalidDestinationException ( const InvalidDestinationException & ex ) throw ()`

6.418.2.3 `cms::InvalidDestinationException::InvalidDestinationException ( const std::string & message, const std::exception * cause ) throw ()`

6.418.2.4 `cms::InvalidDestinationException::InvalidDestinationException ( const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace ) throw ()`

6.418.2.5 `virtual cms::InvalidDestinationException::~InvalidDestinationException ( ) throw ()`  
[virtual]

The documentation for this class was generated from the following file:

- `src/main/cms/InvalidDestinationException.h`

## 6.419 decaf::security::InvalidKeyException Class Reference

```
#include <src/main/decaf/security/InvalidKeyException.h>
```

Inheritance diagram for decaf::security::InvalidKeyException:

### Public Member Functions

- **InvalidKeyException** ( ) throw ()

*Default Constructor.*

- **InvalidKeyException** (const Exception &ex) throw ()

*Conversion Constructor from some other Exception.*

- **InvalidKeyException** (const **InvalidKeyException** &ex) throw ()

*Copy Constructor.*

- **InvalidKeyException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()

*Constructor - Initializes the file name and line number where this message occurred.*

- **InvalidKeyException** (const std::exception \***cause**) throw ()  
*Constructor.*
- **InvalidKeyException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **InvalidKeyException** \* **clone** () const  
*Clones this exception.*
- virtual ~**InvalidKeyException** () throw ()

### 6.419.1 Constructor & Destructor Documentation

6.419.1.1 **decaf::security::InvalidKeyException::InvalidKeyException ( ) throw ()** [inline]

Default Constructor.

6.419.1.2 **decaf::security::InvalidKeyException::InvalidKeyException ( const Exception & ex ) throw ()** [inline]

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.419.1.3 **decaf::security::InvalidKeyException::InvalidKeyException ( const InvalidKeyException & ex ) throw ()** [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.419.1.4 **decaf::security::InvalidKeyException::InvalidKeyException ( const char \* file, const int lineNumber, const std::exception \* cause, const char \* msg, ... ) throw ()** [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.419.1.5** `decaf::security::InvalidKeyException::InvalidKeyException ( const std::exception * cause ) throw () [inline]`

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.419.1.6** `decaf::security::InvalidKeyException::InvalidKeyException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
...	list of primitives that are formatted into the message

**6.419.1.7** `virtual decaf::security::InvalidKeyException::~~InvalidKeyException ( ) throw () [inline, virtual]`

**6.419.2 Member Function Documentation**

**6.419.2.1** `virtual InvalidKeyException* decaf::security::InvalidKeyException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns**

A deep copy of this exception.

Reimplemented from **decaf::security::KeyException** (p. 2368).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**InvalidKeyException.h**

## 6.420 decaf::nio::InvalidMarkException Class Reference

```
#include <src/main/decaf/nio/InvalidMarkException.h>
```

Inheritance diagram for decaf::nio::InvalidMarkException:

### Public Member Functions

- **InvalidMarkException** () throw ()  
*Default Constructor.*
- **InvalidMarkException** (const **lang::Exception** &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **InvalidMarkException** (const **InvalidMarkException** &ex) throw ()  
*Copy Constructor.*
- **InvalidMarkException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **InvalidMarkException** (const std::exception \*cause) throw ()  
*Constructor.*
- **InvalidMarkException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **InvalidMarkException** \* **clone** () const  
*Clones this exception.*
- virtual ~**InvalidMarkException** () throw ()

### 6.420.1 Constructor & Destructor Documentation

#### 6.420.1.1 decaf::nio::InvalidMarkException::InvalidMarkException ( ) throw () [inline]

Default Constructor.

**6.420.1.2** `decaf::nio::InvalidMarkException::InvalidMarkException ( const lang::Exception & ex ) throw () [inline]`

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	The Exception whose state data is to be copied into this Exception.
-----------	---

**6.420.1.3** `decaf::nio::InvalidMarkException::InvalidMarkException ( const InvalidMarkException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	The Exception whose state data is to be copied into this Exception.
-----------	---

**6.420.1.4** `decaf::nio::InvalidMarkException::InvalidMarkException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.420.1.5** `decaf::nio::InvalidMarkException::InvalidMarkException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.420.1.6** `decaf::nio::InvalidMarkException::InvalidMarkException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.  
Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.420.1.7** `virtual decaf::nio::InvalidMarkException::~~InvalidMarkException ( ) throw ()`  
`[inline, virtual]`

### 6.420.2 Member Function Documentation

**6.420.2.1** `virtual InvalidMarkException* decaf::nio::InvalidMarkException::clone ( ) const`  
`[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

A new Exception instance that is a copy of this Exception.

Reimplemented from **decaf::lang::exceptions::IllegalStateException** (p. 2062).

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/InvalidMarkException.h`

## 6.421 cms::InvalidSelectorException Class Reference

This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

```
#include <src/main/cms/InvalidSelectorException.h>
```

Inheritance diagram for cms::InvalidSelectorException:



## Public Member Functions

- **InvalidSelectorException** () throw ()
- **InvalidSelectorException** (const **InvalidSelectorException** &ex) throw ()
- **InvalidSelectorException** (const std::string &message, const std::exception \*cause) throw ()
- **InvalidSelectorException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**InvalidSelectorException** () throw ()

### 6.421.1 Detailed Description

This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

Since

1.3

### 6.421.2 Constructor & Destructor Documentation

6.421.2.1 **cms::InvalidSelectorException::InvalidSelectorException** ( ) throw ()

6.421.2.2 **cms::InvalidSelectorException::InvalidSelectorException** ( const **InvalidSelectorException** & ex ) throw ()

6.421.2.3 **cms::InvalidSelectorException::InvalidSelectorException** ( const std::string & message, const std::exception \* cause ) throw ()

6.421.2.4 **cms::InvalidSelectorException::InvalidSelectorException** ( const std::string & message, const std::exception \* cause, const std::vector< std::pair< std::string, int > > & stackTrace ) throw ()

6.421.2.5 virtual **cms::InvalidSelectorException::~~InvalidSelectorException** ( ) throw ()  
[virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**InvalidSelectorException.h**

## 6.422 decaf::lang::exceptions::InvalidStateException Class Reference

```
#include <src/main/decaf/lang/exceptions/InvalidStateException.h>
```

Inheritance diagram for decaf::lang::exceptions::InvalidStateException:

## Public Member Functions

- **InvalidStateException** () throw ()  
*Default Constructor.*
- **InvalidStateException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1886).*
- **InvalidStateException** (const **InvalidStateException** &ex) throw ()  
*Copy Constructor.*
- **InvalidStateException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **InvalidStateException** (const std::exception \*cause) throw ()  
*Constructor.*
- **InvalidStateException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **InvalidStateException** \* **clone** () const  
*Clones this exception.*
- virtual ~**InvalidStateException** () throw ()

### 6.422.1 Constructor & Destructor Documentation

6.422.1.1 **decaf::lang::exceptions::InvalidStateException::InvalidStateException ( ) throw ()**  
[inline]

Default Constructor.

6.422.1.2 **decaf::lang::exceptions::InvalidStateException::InvalidStateException ( const **Exception** & ex ) throw ()** [inline]

Conversion Constructor from some other **Exception** (p. 1886).

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

**6.422.1.3** decaf::lang::exceptions::InvalidStateException::InvalidStateException ( const InvalidStateException & ex ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

**6.422.1.4** decaf::lang::exceptions::InvalidStateException::InvalidStateException ( const char \* file, const int lineNumber, const std::exception \* cause, const char \* msg, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.  
Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.422.1.5** decaf::lang::exceptions::InvalidStateException::InvalidStateException ( const std::exception \* cause ) throw () [inline]

Constructor.

#### Parameters

<i>cause</i>	<b>Pointer</b> (p. 3032) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

**6.422.1.6** decaf::lang::exceptions::InvalidStateException::InvalidStateException ( const char \* file, const int lineNumber, const char \* msg, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.  
Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.422.1.7 virtual `decaf::lang::exceptions::InvalidStateException::~~InvalidStateException ( )`  
`throw ()` [`inline`, `virtual`]

## 6.422.2 Member Function Documentation

6.422.2.1 virtual `InvalidStateException* decaf::lang::exceptions::InvalidStateException::clone ( )` `const`  
`[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

an new **Exception** (p. 1886) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1889).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/InvalidStateException.h`

## 6.423 decaf::io::IOException Class Reference

```
#include <src/main/decaf/io/IOException.h>
```

Inheritance diagram for `decaf::io::IOException`:

### Public Member Functions

- **IOException ()** `throw ()`  
*Default Constructor.*
- **IOException (const lang::Exception &ex)** `throw ()`  
*Copy Constructor.*
- **IOException (const IOException &ex)** `throw ()`  
*Copy Constructor.*
- **IOException (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...)** `throw ()`  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IOException (const std::exception \*cause)** `throw ()`

*Constructor.*

- **IOException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()

*Constructor.*

- virtual **IOException** \* **clone** () const

*Clones this exception.*

- virtual ~**IOException** () throw ()

### 6.423.1 Constructor & Destructor Documentation

#### 6.423.1.1 decaf::io::IOException::IOException ( ) throw () [inline]

Default Constructor.

#### 6.423.1.2 decaf::io::IOException::IOException ( const lang::Exception & ex ) throw () [inline]

Copy Constructor.

##### Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

#### 6.423.1.3 decaf::io::IOException::IOException ( const IOException & ex ) throw () [inline]

Copy Constructor.

##### Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

#### 6.423.1.4 decaf::io::IOException::IOException ( const char \* file, const int lineNumber, const std::exception \* cause, const char \* msg, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

##### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.

<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.423.1.5** `decaf::io::IOException::IOException ( const std::exception * cause ) throw ()`  
`[inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.423.1.6** `decaf::io::IOException::IOException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()` `[inline]`

Constructor.

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.423.1.7** `virtual decaf::io::IOException::~~IOException ( ) throw ()` `[inline, virtual]`

## 6.423.2 Member Function Documentation

**6.423.2.1** `virtual IOException* decaf::io::IOException::clone ( ) const` `[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

a new instance of an Exception that is a copy of this instance.

Reimplemented from `decaf::lang::Exception` (p. 1889).

Reimplemented in `decaf::internal::net::ssl::openssl::OpenSSLSocketException` (p. 2958), `decaf::io::EOFException` (p. 1883), `decaf::io::InterruptedIOException` (p. 2198),

**decaf::io::UnsupportedEncodingException** (p. 4027), **decaf::io::UTFDataFormatException** (p. 4080), **decaf::net::BindException** (p. 844), **decaf::net::ConnectException** (p. 1296), **decaf::net::HttpRetryException** (p. 2051), **decaf::net::MalformedURLException** (p. 2538), **decaf::net::NoRouteToHostException** (p. 2907), **decaf::net::PortUnreachableException** (p. 3062), **decaf::net::ProtocolException** (p. 3230), **decaf::net::SocketException** (p. 3629), **decaf::net::SocketTimeoutException** (p. 3652), **decaf::net::UnknownHostException** (p. 4022), **decaf::net::UnknownServiceException** (p. 4024), and **decaf::util::zip::ZipException** (p. 4178).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/IOException.h`

## 6.424 activemq::transport::IOTransport Class Reference

Implementation of the **Transport** (p. 3996) interface that performs marshaling of commands to IO streams.

```
#include <src/main/activemq/transport/IOTransport.h>
```

Inheritance diagram for `activemq::transport::IOTransport`:

### Public Member Functions

- **IOTransport** ()

*Default Constructor.*

- **IOTransport** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)

*Create an instance of this **Transport** (p. 3996) and assign its **WireFormat** instance at creation time.*

- virtual **~IOTransport** ()
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw ( **decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException** )

*Sends a one-way command.*

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw ( **decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException** )

*Not supported by this class - throws an exception.*

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw ( **decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException** )

*Not supported by this class - throws an exception.*

- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wire-Format)  
*Sets the WireFormat instance to use.*
- virtual void **setTransportListener** (**TransportListener** \*listener)  
*Sets the observer of asynchronous exceptions from this transport.*
- virtual **TransportListener** \* **getTransportListener** () const  
*Gets the observer of asynchronous exceptions from this transport.*
- virtual void **setInputStream** (**decaf::io::DataInputStream** \*is)  
*Sets the input stream for in-coming commands.*
- virtual void **setOutputStream** (**decaf::io::DataOutputStream** \*os)  
*Sets the output stream for out-going commands.*
- virtual void **start** () throw ( **decaf::io::IOException** )  
*Starts this transport object and creates the thread for polling on the input stream for commands.*
- virtual void **stop** () throw ( **decaf::io::IOException** )  
*Stops the **Transport** (p.3996), terminating any threads and stopping all read and write operations.*
- virtual void **close** () throw ( **decaf::io::IOException** )  
*Stops the polling thread and closes the streams.*
- virtual void **run** ()  
*Runs the polling thread.*
- virtual **Transport** \* **narrow** (const std::type\_info &typeId)  
*Narrows down a Chain of Transports to a specific **Transport** (p.3996) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () const  
*Is this **Transport** (p.3996) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const  
*Is the **Transport** (p.3996) Connected to its Broker.*
- virtual bool **isClosed** () const  
*Has the **Transport** (p.3996) been shutdown and no longer usable.*
- virtual std::string **getRemoteAddress** () const



- virtual void **reconnect** (const **decaf::net::URI** &uri AMQCPP\_UNUSED) throw ( decaf::io::IOException )  
*reconnect to another location*

### 6.424.1 Detailed Description

Implementation of the **Transport** (p. 3996) interface that performs marshaling of commands to IO streams. This class does not implement the request method, it only handles oneway messages. A thread polls on the input stream for in-coming commands. When a command is received, the command listener is notified. The polling thread is not started until the start method is called. The close method will close the associated streams. Close can be called explicitly by the user, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

### 6.424.2 Constructor & Destructor Documentation

#### 6.424.2.1 activemq::transport::IOTransport::IOTransport ( )

Default Constructor.

#### 6.424.2.2 activemq::transport::IOTransport::IOTransport ( const Pointer< wireformat::WireFormat > & wireFormat )

Create an instance of this **Transport** (p. 3996) and assign its WireFormat instance at creation time.

#### Parameters

<i>wireFormat</i>	Data encoder / decoder to use when reading and writing.
-------------------	---

#### 6.424.2.3 virtual activemq::transport::IOTransport::~~IOTransport ( ) [virtual]

### 6.424.3 Member Function Documentation

#### 6.424.3.1 virtual void activemq::transport::IOTransport::close ( ) throw ( decaf::io::IOException ) [virtual]

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

#### Exceptions

<i>IOException</i>	if errors occur.
--------------------	------------------

Implements **decaf::io::Closeable** (p. 1181).

**6.424.3.2** `virtual std::string activemq::transport::IOTransport::getRemoteAddress ( ) const`  
`[inline, virtual]`

#### Returns

the remote address for this connection

Implements **activemq::transport::Transport** (p. 3998).

**6.424.3.3** `virtual TransportListener* activemq::transport::IOTransport::getTransportListener`  
`( ) const [inline, virtual]`

Gets the observer of asynchronous exceptions from this transport.

#### Returns

The listener of transport events.

Implements **activemq::transport::Transport** (p. 3998).

**6.424.3.4** `virtual bool activemq::transport::IOTransport::isClosed ( ) const` `[inline,`  
`virtual]`

Has the **Transport** (p. 3996) been shutdown and no longer usable.

#### Returns

true if the **Transport** (p. 3996)

Implements **activemq::transport::Transport** (p. 3998).

**6.424.3.5** `virtual bool activemq::transport::IOTransport::isConnected ( ) const` `[inline,`  
`virtual]`

Is the **Transport** (p. 3996) Connected to its Broker.

#### Returns

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 3998).

**6.424.3.6** `virtual bool activemq::transport::IOTransport::isFaultTolerant ( ) const`  
`[inline, virtual]`

Is this **Transport** (p. 3996) fault tolerant, meaning that it will reconnect to a broker on disconnect.

#### Returns

true if the **Transport** (p. 3996) is fault tolerant.

Implements **activemq::transport::Transport** (p. 3999).

**6.424.3.7** `virtual Transport* activemq::transport::IOTransport::narrow ( const std::type_info & typeId )`  
`[inline, virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 3996) to allow a higher level transport to skip intermediate Transports in certain circumstances.

#### Parameters

<i>typeId</i>	- The type_info of the Object we are searching for.
---------------	---

#### Returns

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 3999).

**6.424.3.8** `virtual void activemq::transport::IOTransport::oneway ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )`  
`[virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

#### Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

#### Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 3999).

**6.424.3.9** `virtual void activemq::transport::IOTransport::reconnect ( const decaf::net::URI &uri AMQCPP_UNUSED ) throw ( decaf::io::IOException ) [inline, virtual]`

reconnect to another location

#### Parameters

<i>uri</i>	
------------	--

#### Exceptions

<i>IOException</i>	on failure of if not supported
--------------------	--------------------------------

**6.424.3.10** `virtual Pointer<Response> activemq::transport::IOTransport::request ( const Pointer< Command > &command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Not supported by this class - throws an exception.

#### Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

#### Returns

the response to the command sent.

#### Exceptions

<i>UnsupportedOperationException.</i>	
---------------------------------------	--

Implements `activemq::transport::Transport` (p. 4000).

**6.424.3.11** `virtual Pointer<Response> activemq::transport::IOTransport::request ( const Pointer< Command > &command, unsigned int timeout ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Not supported by this class - throws an exception.

#### Parameters

<i>command</i>	the command to be sent.
<i>timeout</i>	the time to wait for a response.

**Returns**

the response to the command sent.

**Exceptions**

<i>UnsupportedOperation Exception.</i>	
--	--

Implements **activemq::transport::Transport** (p. 4001).

**6.424.3.12** `virtual void activemq::transport::IOTransport::run ( ) [virtual]`

Runs the polling thread.

Implements **decaf::lang::Runnable** (p. 3419).

**6.424.3.13** `virtual void activemq::transport::IOTransport::setInputStream ( decaf::io::DataInputStream * is ) [inline, virtual]`

Sets the input stream for in-coming commands.

**Parameters**

<i>is</i>	The input stream.
-----------	-------------------

**6.424.3.14** `virtual void activemq::transport::IOTransport::setOutputStream ( decaf::io::DataOutputStream * os ) [inline, virtual]`

Sets the output stream for out-going commands.

**Parameters**

<i>os</i>	The output stream.
-----------	--------------------

**6.424.3.15** `virtual void activemq::transport::IOTransport::setTransportListener ( TransportListener * listener ) [inline, virtual]`

Sets the observer of asynchronous exceptions from this transport.

**Parameters**

<i>listener</i>	the listener of transport events.
-----------------	-----------------------------------

Implements **activemq::transport::Transport** (p. 4001).

**6.424.3.16** `virtual void activemq::transport::IOTransport::setWireFormat ( const Pointer< wireformat::WireFormat > & wireFormat ) [inline, virtual]`

Sets the WireFormat instance to use.

#### Parameters

<i>wireFormat</i>	The WireFormat the object used to encode / decode commands.
-------------------	---

Implements **activemq::transport::Transport** (p. 4002).

**6.424.3.17** `virtual void activemq::transport::IOTransport::start ( ) throw ( decaf::io::IOException ) [virtual]`

Starts this transport object and creates the thread for polling on the input stream for commands.

If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

#### Exceptions

<i>CMSEException</i>	if an error occurs or if this transport has already been closed.
----------------------	--

Implements **activemq::transport::Transport** (p. 4002).

**6.424.3.18** `virtual void activemq::transport::IOTransport::stop ( ) throw ( decaf::io::IOException ) [virtual]`

Stops the **Transport** (p. 3996), terminating any threads and stopping all read and write operations.

#### Exceptions

<i>IOException</i>	if an error occurs while stopping the <b>Transport</b> (p. 3996).
--------------------	---

Implements **activemq::transport::Transport** (p. 4002).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/IOTransport.h`

## 6.425 decaf::lang::Iterable< E > Class Template Reference

Implementing this interface allows an object to be cast to an **Iterable** (p. 2220) type for generic collections API calls.

```
#include <src/main/decaf/lang/Iterable.h>
```

Inheritance diagram for decaf::lang::Iterable< E >:

## Public Member Functions

- virtual **~Iterable** ()
- virtual **decaf::util::Iterator**< E > \* **iterator** ()=0
- virtual **decaf::util::Iterator**< E > \* **iterator** () const =0

### 6.425.1 Detailed Description

template<typename E> class decaf::lang::Iterable< E >

Implementing this interface allows an object to be cast to an **Iterable** (p. 2220) type for generic collections API calls.

### 6.425.2 Constructor & Destructor Documentation

6.425.2.1 template<typename E> virtual decaf::lang::Iterable< E >::~~Iterable ( )  
[inline, virtual]

### 6.425.3 Member Function Documentation

6.425.3.1 template<typename E> virtual decaf::util::Iterator<E>\*  
decaf::lang::Iterable< E >::iterator ( ) [pure virtual]

## Returns

an iterator over a set of elements of type T.

Implemented in **decaf::util::concurrent::SynchronousQueue**< E > (p. 3833), **decaf::util::PriorityQueue**< E > (p. 3121), **decaf::util::StlList**< E > (p. 3704), **decaf::util::StlSet**< E > (p. 3736), **decaf::util::StlList**< cms::MessageConsumer \* > (p. 3704), **decaf::util::StlList**< CompositeTask \* > (p. 3704), **decaf::util::StlList**< URI > (p. 3704), **decaf::util::StlList**< Pointer< DestinationInfo > > (p. 3704), **decaf::util::StlList**< PrimitiveValueNode > (p. 3704), **decaf::util::StlList**< Pointer< Command > > (p. 3704), **decaf::util::StlList**< Pointer< BackupTransport > > (p. 3704), **decaf::util::StlList**< cms::MessageProducer \* > (p. 3704), **decaf::util::StlList**< cms::Destination \* > (p. 3704), **decaf::util::StlList**< cms::Session \* > (p. 3704), **decaf::util::StlList**< cms::Connection \* > (p. 3704), **decaf::util::StlSet**< transport::TransportListener \* > (p. 3736), **decaf::util::StlSet**< Pointer< Synchronization > > (p. 3736), **decaf::util::StlSet**< Resource \* > (p. 3736), and **decaf::util::StlSet**< ActiveMQSession \* > (p. 3736).

Referenced by **decaf::util::AbstractCollection**< cms::Connection \* >::clear(), **decaf::util::AbstractCollection**< cms::Connection \* >::contains(), **decaf::util::AbstractCollection**< cms::Connection \* >::copy(), **decaf::util::AbstractCollection**< cms::Connection \* >::operator=(), **decaf::util::AbstractCollection**< cms::Connection \* >::remove(), **decaf::util::AbstractSet**< ActiveMQSession \* >::removeAll(),

decaf::util::AbstractCollection< cms::Connection \* >::removeAll(), decaf::util::AbstractCollection< cms::Connection \* >::retainAll(), and decaf::util::AbstractCollection< cms::Connection \* >::toArray().

**6.425.3.2** `template<typename E> virtual decaf::util::Iterator<E>*`  
`decaf::lang::Iterable< E >::iterator ( ) const` [pure virtual]

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3833), `decaf::util::PriorityQueue< E >` (p. 3121), `decaf::util::StlList< E >` (p. 3704), `decaf::util::StlSet< E >` (p. 3736), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3704), `decaf::util::StlList< CompositeTask * >` (p. 3704), `decaf::util::StlList< URI >` (p. 3704), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3704), `decaf::util::StlList< PrimitiveValueNode >` (p. 3704), `decaf::util::StlList< Pointer< Command > >` (p. 3704), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3704), `decaf::util::StlList< cms::MessageProducer * >` (p. 3704), `decaf::util::StlList< cms::Destination * >` (p. 3704), `decaf::util::StlList< cms::Session * >` (p. 3704), `decaf::util::StlList< cms::Connection * >` (p. 3704), `decaf::util::StlSet< transport::TransportListener * >` (p. 3736), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 3736), `decaf::util::StlSet< Resource * >` (p. 3736), and `decaf::util::StlSet< ActiveMQSession * >` (p. 3736).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Iterable.h`

## 6.426 decaf::util::Iterator< T > Class Template Reference

Defines an object that can be used to iterate over the elements of a collection.

```
#include <src/main/decaf/util/Iterator.h>
```

Inheritance diagram for `decaf::util::Iterator< T >`:

### Public Member Functions

- virtual `~Iterator ()`
- virtual `T next ()=0` throw ( `lang::exceptions::NoSuchElementException` )  
*Returns the next element in the iteration.*
- virtual bool `hasNext ()` const =0  
*Returns true if the iteration has more elements.*
- virtual void `remove ()=0` throw ( `lang::exceptions::IllegalStateException`, `lang::exceptions::UnsupportedOperationException` )  
*Removes from the underlying collection the last element returned by the iterator (optional operation).*



### 6.426.1 Detailed Description

`template<typename T> class decaf::util::Iterator< T >`

Defines an object that can be used to iterate over the elements of a collection. The iterator provides a way to access and remove elements with well defined semantics.

### 6.426.2 Constructor & Destructor Documentation

**6.426.2.1** `template<typename T> virtual decaf::util::Iterator< T >::~~Iterator ( )`  
`[inline, virtual]`

### 6.426.3 Member Function Documentation

**6.426.3.1** `template<typename T> virtual bool decaf::util::Iterator< T >::hasNext ( )`  
`const [pure virtual]`

Returns true if the iteration has more elements.

Returns false if the next call to next would result in an NoSuchElementException to be thrown.

**6.426.3.2** `template<typename T> virtual T decaf::util::Iterator< T >::next ( ) throw (`  
`lang::exceptions::NoSuchElementException) [pure virtual]`

Returns the next element in the iteration.

Calling this method repeatedly until the **hasNext()** (p. 2223) method returns false will return each element in the underlying collection exactly once.

#### Returns

next element in the iteration of elements

#### Exceptions

<i>NoSuchElementException</i>	- iteration has no more elements.
-------------------------------	-----------------------------------

**6.426.3.3** `template<typename T> virtual void decaf::util::Iterator< T`  
`>::remove ( ) throw ( lang::exceptions::IllegalStateException,`  
`lang::exceptions::UnsupportedOperationException) [pure`  
`virtual]`

Removes from the underlying collection the last element returned by the iterator (optional operation).

This method can be called only once per call to next. The behavior of an iterator is unspecified if the underlying collection is modified while the iteration is in progress in

any way other than by calling this method.

### Exceptions

<i>UnsupportedOperationException</i>	- if the remove operation is not supported by this <b>Iterator</b> (p. 2222).
<i>IllegalStateException</i>	- if the next method has not yet been called, or the remove method has already been called after the last call to the next method.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Iterator.h**

## 6.427 activemq::commands::JournalQueueAck Class Reference

```
#include <src/main/activemq/commands/JournalQueueAck.h>
```

Inheritance diagram for activemq::commands::JournalQueueAck:

### Public Member Functions

- **JournalQueueAck** ()
- virtual **~JournalQueueAck** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **JournalQueueAck \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()

- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &**destination**)
- virtual const **Pointer**< **MessageAck** > &**getMessageAck** () const
- virtual **Pointer**< **MessageAck** > &**getMessageAck** ()
- virtual void **setMessageAck** (const **Pointer**< **MessageAck** > &**messageAck**)

### Static Public Attributes

- static const unsigned char **ID\_JOURNALQUEUEACK** = 52

### Protected Attributes

- **Pointer**< **ActiveMQDestination** > **destination**
- **Pointer**< **MessageAck** > **messageAck**

## 6.427.1 Constructor & Destructor Documentation

6.427.1.1 **activemq::commands::JournalQueueAck::JournalQueueAck** ( )

6.427.1.2 **virtual activemq::commands::JournalQueueAck::~~JournalQueueAck** ( )  
[virtual]

## 6.427.2 Member Function Documentation

6.427.2.1 **virtual JournalQueueAck\* activemq::commands::JournalQueueAck::cloneDataStructure**  
( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

6.427.2.2 **virtual void activemq::commands::JournalQueueAck::copyDataStructure** ( const  
**DataStructure** \* **src** ) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<i>src</i>	- Source Object
------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1715).

**6.427.2.3** `virtual bool activemq::commands::JournalQueueAck::equals ( const DataStructure  
* value ) const` [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1716).

**6.427.2.4** `virtual unsigned char activemq::commands::JournalQueueAck::getDataStructureType  
( ) const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

- 6.427.2.5 `virtual const Pointer<ActiveMQDestination>& activemq::commands::JournalQueueAck::getDestination ( ) const` [virtual]
- 6.427.2.6 `virtual Pointer<ActiveMQDestination>& activemq::commands::JournalQueueAck::getDestination ( )` [virtual]
- 6.427.2.7 `virtual const Pointer<MessageAck>& activemq::commands::JournalQueueAck::getMessageAck ( ) const` [virtual]
- 6.427.2.8 `virtual Pointer<MessageAck>& activemq::commands::JournalQueueAck::getMessageAck ( )` [virtual]
- 6.427.2.9 `virtual void activemq::commands::JournalQueueAck::setDestination ( const Pointer< ActiveMQDestination > & destination )` [virtual]
- 6.427.2.10 `virtual void activemq::commands::JournalQueueAck::setMessageAck ( const Pointer< MessageAck > & messageAck )` [virtual]
- 6.427.2.11 `virtual std::string activemq::commands::JournalQueueAck::toString ( ) const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 841).

### 6.427.3 Field Documentation

- 6.427.3.1 `Pointer<ActiveMQDestination> activemq::commands::JournalQueueAck::destination` [protected]
- 6.427.3.2 `const unsigned char activemq::commands::JournalQueueAck::ID_-JOURNALQUEUEACK = 52` [static]
- 6.427.3.3 `Pointer<MessageAck> activemq::commands::JournalQueueAck::messageAck` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalQueueAck.h`

## 6.428 activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2228).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/JournalQueueAck
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller`:

### Public Member Functions

- **JournalQueueAckMarshaller ()**
- virtual **~JournalQueueAckMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn)** throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

6.428

**activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller**

Class Reference

2231

### 6.428.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2228).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.428.2 Constructor & Destructor Documentation

6.428.2.1 **activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::JournalQueueAckMarshaller**  
( ) [inline]

6.428.2.2 **virtual activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller**  
( ) [inline, virtual]

### 6.428.3 Member Function Documentation

6.428.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.428.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.428.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.428.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.428.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.



6.428

**activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller****Class Reference****2233****Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.428.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.428.3.7** `virtual void activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**JournalQueueAckMarshaller.h**

## 6.429 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2232).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAck
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller:

### Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual  $\sim$ **JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

6.429

**activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller**

**Class Reference**

2235

---

*Write a object instance to data output stream.*

### 6.429.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2232).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.429.2 Constructor & Destructor Documentation

6.429.2.1 **activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::JournalQueueAckMarshaller**  
( ) [inline]

6.429.2.2 **virtual activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller**  
( ) [inline, virtual]

### 6.429.3 Member Function Documentation

6.429.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.429.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.429.3.3 **virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.429.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.429.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

6.429

**activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller****Class Reference****2237****Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.429.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.429.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAckMarshaller.h`

## 6.430 `activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2236).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAck
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller`:

### Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

6.430

**activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller**

**Class Reference**

2239

---

*Write a object instance to data output stream.*

### 6.430.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2236).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.430.2 Constructor & Destructor Documentation

6.430.2.1 **activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::JournalQueueAckMarshaller**  
( ) [inline]

6.430.2.2 **virtual activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller**  
( ) [inline, virtual]

### 6.430.3 Member Function Documentation

6.430.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.430.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.430.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.430.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.430.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.



## 6.430

**activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller**

**Class Reference**

**2241**

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.430.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.430.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h`

## 6.431 `activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller` Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2240).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAck
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller`:

### Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

---

*Write a object instance to data output stream.*

### 6.431.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2240).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.431.2 Constructor & Destructor Documentation

6.431.2.1 **activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::JournalQueueAckMarshaller**  
( ) [inline]

6.431.2.2 **virtual activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller**  
( ) [inline, virtual]

### 6.431.3 Member Function Documentation

6.431.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.431.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.431.3.3 **virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.431.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.431.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

6.431

**activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller****Class Reference****2245****Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.431.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.431.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h`

## 6.432 `activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2244).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAck
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller`:

### Public Member Functions

- **JournalQueueAckMarshaller ()**
- virtual **~JournalQueueAckMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)**  
**throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** **throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs)** **throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn)** **throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut)** **throw ( decaf::io::IOException )**

6.432

**activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller**

**Class Reference**

2247

---

*Write a object instance to data output stream.*

### 6.432.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2244).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.432.2 Constructor & Destructor Documentation

6.432.2.1 **activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::JournalQueueAckMarshaller**  
( ) [inline]

6.432.2.2 **virtual activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller**  
( ) [inline, virtual]

### 6.432.3 Member Function Documentation

6.432.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.432.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.432.3.3 **virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.432.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.432.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.



6.432

**activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller**

**Class Reference**

**2249**

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.432.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.432.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller.h`

## 6.433 `activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller` Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2248).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalQueueAck
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller`:

### Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

6.433

**activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller**

**Class Reference**

2251

---

*Write a object instance to data output stream.*

### 6.433.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2248).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.433.2 Constructor & Destructor Documentation

6.433.2.1 **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::JournalQueueAckMarshaller**  
( ) [inline]

6.433.2.2 **virtual activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller**  
( ) [inline, virtual]

### 6.433.3 Member Function Documentation

6.433.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.433.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.433.3.3 **virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.433.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.433.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

6.433

**activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller****Class Reference****2253****Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.433.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.433.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshall/v1/**JournalQueueAckMarshaller.h**

## 6.434 activemq::commands::JournalTopicAck Class Reference

```
#include <src/main/activemq/commands/JournalTopicAck.h>
```

Inheritance diagram for activemq::commands::JournalTopicAck:

### Public Member Functions

- **JournalTopicAck ()**
- virtual **~JournalTopicAck ()**
- virtual unsigned char **getDataStructureType ()** const  
*Get the unique identifier that this object and its own Marshaller share.*
- virtual **JournalTopicAck \* cloneDataStructure ()** const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure (const DataStructure \*src)**  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString ()** const  
*Returns a string containing the information for this DataStructure (p. 1713) such as its type and value of its elements.*
- virtual bool **equals (const DataStructure \*value)** const  
*Compares the DataStructure (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer< ActiveMQDestination > & getDestination ()** const
- virtual **Pointer< ActiveMQDestination > & getDestination ()**
- virtual void **setDestination (const Pointer< ActiveMQDestination > &destination)**
- virtual const **Pointer< MessageId > & getMessageId ()** const
- virtual **Pointer< MessageId > & getMessageId ()**
- virtual void **setMessageId (const Pointer< MessageId > &messageId)**
- virtual long long **getMessageSequenceId ()** const
- virtual void **setMessageSequenceId (long long messageSequenceId)**
- virtual const std::string & **getSubscriptionName ()** const
- virtual std::string & **getSubscriptionName ()**
- virtual void **setSubscriptionName (const std::string &subscriptionName)**
- virtual const std::string & **getClientId ()** const

- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)

### Static Public Attributes

- static const unsigned char **ID\_JOURNALTOPICACK** = 50

### Protected Attributes

- **Pointer**< **ActiveMQDestination** > destination
- **Pointer**< **MessageId** > messageId
- long long **messageSequenceId**
- std::string **subscriptionName**
- std::string **clientId**
- **Pointer**< **TransactionId** > transactionId

## 6.434.1 Constructor & Destructor Documentation

6.434.1.1 **activemq::commands::JournalTopicAck::JournalTopicAck** ( )

6.434.1.2 **virtual activemq::commands::JournalTopicAck::~~JournalTopicAck** ( )  
[virtual]

## 6.434.2 Member Function Documentation

6.434.2.1 **virtual JournalTopicAck\* activemq::commands::JournalTopicAck::cloneDataStructure**  
( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

6.434.2.2 **virtual void activemq::commands::JournalTopicAck::copyDataStructure** ( const  
**DataStructure** \* src ) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

**Parameters**

<i>src</i>	- Source Object
------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1715).

**6.434.2.3** `virtual bool activemq::commands::JournalTopicAck::equals ( const DataStructure  
* value ) const` [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns**

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1716).

**6.434.2.4** `virtual const std::string& activemq::commands::JournalTopicAck::getClientId ( )  
const` [virtual]

**6.434.2.5** `virtual std::string& activemq::commands::JournalTopicAck::getClientId ( )  
[virtual]`

**6.434.2.6** `virtual unsigned char activemq::commands::JournalTopicAck::getDataStructureType ( ) const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns**

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).



- 6.434.2.7 `virtual const Pointer<ActiveMQDestination>& activemq::commands::JournalTopicAck::getDestination ( ) const [virtual]`
- 6.434.2.8 `virtual Pointer<ActiveMQDestination>& activemq::commands::JournalTopicAck::getDestination ( ) [virtual]`
- 6.434.2.9 `virtual const Pointer<MessageId>& activemq::commands::JournalTopicAck::getMessageId ( ) const [virtual]`
- 6.434.2.10 `virtual Pointer<MessageId>& activemq::commands::JournalTopicAck::getMessageId ( ) [virtual]`
- 6.434.2.11 `virtual long long activemq::commands::JournalTopicAck::getMessageSequenceId ( ) const [virtual]`
- 6.434.2.12 `virtual std::string& activemq::commands::JournalTopicAck::getSubscriptionName ( ) [virtual]`
- 6.434.2.13 `virtual const std::string& activemq::commands::JournalTopicAck::getSubscriptionName ( ) const [virtual]`
- 6.434.2.14 `virtual const Pointer<TransactionId>& activemq::commands::JournalTopicAck::getTransactionId ( ) const [virtual]`
- 6.434.2.15 `virtual Pointer<TransactionId>& activemq::commands::JournalTopicAck::getTransactionId ( ) [virtual]`
- 6.434.2.16 `virtual void activemq::commands::JournalTopicAck::setClientId ( const std::string & clientId ) [virtual]`
- 6.434.2.17 `virtual void activemq::commands::JournalTopicAck::setDestination ( const Pointer< ActiveMQDestination > & destination ) [virtual]`
- 6.434.2.18 `virtual void activemq::commands::JournalTopicAck::setMessageId ( const Pointer< MessageId > & messageId ) [virtual]`
- 6.434.2.19 `virtual void activemq::commands::JournalTopicAck::setMessageSequenceId ( long long messageSequenceId ) [virtual]`
- 6.434.2.20 `virtual void activemq::commands::JournalTopicAck::setSubscriptionName ( const std::string & subscriptionName ) [virtual]`
- 6.434.2.21 `virtual void activemq::commands::JournalTopicAck::setTransactionId ( const Pointer< TransactionId > & transactionId ) [virtual]`
- 6.434.2.22 `virtual std::string activemq::commands::JournalTopicAck::toString ( ) const [virtual]`

---

Generated on Tue Mar 8 2011 17:10:05 for activemq-cpp-3.2.5 by Doxygen

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

**Returns**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 841).

**6.434.3 Field Documentation**

**6.434.3.1** `std::string activemq::commands::JournalTopicAck::clientId`  
[protected]

**6.434.3.2** `Pointer<ActiveMQDestination> activemq::commands::JournalTopicAck::destination`  
[protected]

**6.434.3.3** `const unsigned char activemq::commands::JournalTopicAck::ID_-JOURNALTOPICACK = 50` [static]

**6.434.3.4** `Pointer<MessageId> activemq::commands::JournalTopicAck::messageId`  
[protected]

**6.434.3.5** `long long activemq::commands::JournalTopicAck::messageSequenceId`  
[protected]

**6.434.3.6** `std::string activemq::commands::JournalTopicAck::subscriptionName`  
[protected]

**6.434.3.7** `Pointer<TransactionId> activemq::commands::JournalTopicAck::transactionId`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalTopicAck.h`

**6.435 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller Class Reference**

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2256).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalTopicAck
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller`:

- **JournalTopicAckMarshaller ()**
- virtual **~JournalTopicAckMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.435.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2256).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.435.2 Constructor & Destructor Documentation

6.435.2.1 `activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::JournalTopicAckMarshaller ( ) [inline]`

6.435.2.2 `virtual activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller ( ) [inline, virtual]`

## 6.435.3 Member Function Documentation

6.435.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.435.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.435.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

6.435

**activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller****Class Reference****2261****Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.435.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.435.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```

6.435.3.6 virtual void activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```

6.435.3.7 virtual void activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**JournalTopicAckMarshaller.h**

6.436

activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller

Class Reference

6.436 — activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller <sup>2263</sup>

## Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2261).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalTopicAckMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller:

## Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.436.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2261).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.436.2 Constructor & Destructor Documentation

6.436.2.1 `activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::JournalTopicAckMarshaller ( ) [inline]`

6.436.2.2 `virtual activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller ( ) [inline, virtual]`

### 6.436.3 Member Function Documentation

6.436.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.436.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.436.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters



6.436

**activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller****Class Reference****2265**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.436.3.4** virtual void **activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.436.3.5** virtual int **activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::tightMarshal1** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.436.3.6  virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.436.3.7  virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

6.437

activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller

Class Reference

2267

- src/main/activemq/wireformat/openwire/marshal/v2/JournalTopicAckMarshaller.h

## 6.437 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2265).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller:

### Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.437.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2265).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.437.2 Constructor & Destructor Documentation

6.437.2.1 **activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::JournalTopicAckMarshaller**  
 ( ) [inline]

6.437.2.2 **virtual activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller**  
 ( ) [inline, virtual]

### 6.437.3 Member Function Documentation

6.437.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::createObject (**  
**) const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.437.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::getDataStructureType**  
**( ) const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.437.3.3 **virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure**  
**\* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw (**  
**decaf::io::IOException )** [virtual]

Write a object instance to data output stream.

6.437

**activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller****Class Reference****2269****Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.437.3.4** virtual void **activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.437.3.5** virtual int **activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::tightMarshal1** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.437.3.6  virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.437.3.7  virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

6.438

activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller

Class Reference

2271

- src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h

## 6.438 activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2269).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/JournalTopicAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller:

### Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.438.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2269).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.438.2 Constructor & Destructor Documentation

6.438.2.1 **activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::JournalTopicAckMarshaller**  
 ( ) [inline]

6.438.2.2 **virtual activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller**  
 ( ) [inline, virtual]

### 6.438.3 Member Function Documentation

6.438.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::createObject (**  
**) const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.438.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::getDataStructureType**  
**( ) const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.438.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure**  
**\* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw (**  
**decaf::io::IOException)** [virtual]

Write a object instance to data output stream.



6.438

**activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller****Class Reference****2273****Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.438.3.4** virtual void **activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.438.3.5** virtual int **activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::tightMarshal1** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.438.3.6  virtual void activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.438.3.7  virtual void activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

6.439

activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller

Class Reference

2275

- src/main/activemq/wireformat/openwire/marshal/v6/JournalTopicAckMarshaller.h

## 6.439 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2273).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller:

### Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.439.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2273).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.439.2 Constructor & Destructor Documentation

6.439.2.1 **activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::JournalTopicAckMarshaller**  
 ( ) [inline]

6.439.2.2 **virtual activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller**  
 ( ) [inline, virtual]

### 6.439.3 Member Function Documentation

6.439.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::createObject (**  
**) const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.439.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::getDataStructureTy**  
**( ) const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.439.3.3 **virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure**  
**\* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw (**  
**decaf::io::IOException )** [virtual]

Write a object instance to data output stream.

6.439

activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller

Class Reference

2277

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.439.3.4** virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::looseUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.439.3.5** virtual int activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::tightMarshal1 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.439.3.6  virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.439.3.7  virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

6.440

activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller

Class Reference

2279

- src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h

## 6.440 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2277).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller:

### Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.440.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2277).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.440.2 Constructor & Destructor Documentation

6.440.2.1 **activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::JournalTopicAckMarshaller**  
 ( ) [inline]

6.440.2.2 **virtual activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller**  
 ( ) [inline, virtual]

### 6.440.3 Member Function Documentation

6.440.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::createObject (**  
**) const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.440.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::getDataStructureType**  
**( ) const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.440.3.3 **virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure**  
**\* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw (**  
**decaf::io::IOException )** [virtual]

Write a object instance to data output stream.



## 6.440

**activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller**

### Class Reference

2281

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.440.3.4** virtual void **activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.440.3.5** virtual int **activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::tightMarshal1** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.440.3.6  virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.440.3.7  virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshall/v1/**JournalTopicAckMarshaller.h**

## 6.441 activemq::commands::JournalTrace Class Reference

```
#include <src/main/activemq/commands/JournalTrace.h>
```

Inheritance diagram for activemq::commands::JournalTrace:

### Public Member Functions

- **JournalTrace** ()
- virtual **~JournalTrace** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **JournalTrace** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getMessage** () const
- virtual std::string & **getMessage** ()
- virtual void **setMessage** (const std::string &message)

### Static Public Attributes

- static const unsigned char **ID\_JOURNALTRACE** = 53

### Protected Attributes

- std::string **message**

### 6.441.1 Constructor & Destructor Documentation

6.441.1.1 `activemq::commands::JournalTrace::JournalTrace ( )`

6.441.1.2 `virtual activemq::commands::JournalTrace::~~JournalTrace ( ) [virtual]`

### 6.441.2 Member Function Documentation

6.441.2.1 `virtual JournalTrace* activemq::commands::JournalTrace::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1714).

6.441.2.2 `virtual void activemq::commands::JournalTrace::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Implements `activemq::commands::DataStructure` (p. 1715).

6.441.2.3 `virtual bool activemq::commands::JournalTrace::equals ( const DataStructure * value ) const [virtual]`

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements `activemq::commands::DataStructure` (p. 1716).

**6.441.2.4** `virtual unsigned char activemq::commands::JournalTrace::getDataStructureType ( )`  
`const [virtual]`

Get the unique identifier that this object and its own Marshaller share.

#### Returns

new **DataSet** (p. 1713) type copy.

Implements **activemq::commands::DataSet** (p. 1717).

**6.441.2.5** `virtual std::string& activemq::commands::JournalTrace::getMessage ( )`  
`[virtual]`

**6.441.2.6** `virtual const std::string& activemq::commands::JournalTrace::getMessage ( ) const`  
`[virtual]`

**6.441.2.7** `virtual void activemq::commands::JournalTrace::setMessage ( const std::string &`  
`message ) [virtual]`

**6.441.2.8** `virtual std::string activemq::commands::JournalTrace::toString ( ) const`  
`[virtual]`

Returns a string containing the information for this **DataSet** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 841).

### 6.441.3 Field Documentation

**6.441.3.1** `const unsigned char activemq::commands::JournalTrace::ID_-`  
`JOURNALTRACE = 53 [static]`

**6.441.3.2** `std::string activemq::commands::JournalTrace::message`  
`[protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalTrace.h`

## 6.442 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller

### Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2283).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMar
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller:

## Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.442.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2283). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.442.2 Constructor & Destructor Documentation

6.442.2.1 `activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::JournalTraceMarshaller ( ) [inline]`

6.442.2.2 `virtual activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::~~JournalTraceMarshaller ( ) [inline, virtual]`

## 6.442.3 Member Function Documentation

6.442.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.442.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.442.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.442.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.442.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).



6.442.3.6 virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::tightMarshal2  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure  
 \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
 utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
 [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

6.442.3.7 virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::tightUnmarshal  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream  
 \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**JournalTraceMarshaller.h**

## 6.443 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller

### Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2288).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMar
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller:

### Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.443.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2288). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.443.2 Constructor & Destructor Documentation

**6.443.2.1** `activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::JournalTraceMarshaller ( ) [inline]`

**6.443.2.2** `virtual activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::~~JournalTraceMarshaller ( ) [inline, virtual]`

### 6.443.3 Member Function Documentation

**6.443.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.443.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.443.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

```
6.443.3.4 virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

```
6.443.3.5 virtual int activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.443.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.443.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h`

## 6.444 `activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller` Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2292).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/JournalTraceMar
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller`:

### Public Member Functions

- **JournalTraceMarshaller** ()
- virtual  $\sim$ **JournalTraceMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.444.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2292). NOTE!  
This file is auto generated - do not modify! if you need to make a change, please see  
the Java Classes in the activemq-openwire-generator module

### 6.444.2 Constructor & Destructor Documentation

**6.444.2.1** `activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::JournalTraceMarshaller  
( ) [inline]`

**6.444.2.2** `virtual activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::~~JournalTraceMarshaller  
( ) [inline, virtual]`

### 6.444.3 Member Function Documentation

**6.444.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::createObject ( )  
const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.444.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::getDataStructureType  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.444.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::looseMarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (  
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.444.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.444.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.



## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.444.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.444.3.7** `virtual void activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/JournalTraceMarshaller.h`

## 6.445 `activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2296).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMar
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller`:

### Public Member Functions

- **JournalTraceMarshaller** ()
- virtual  $\sim$ **JournalTraceMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.445.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2296). NOTE!  
This file is auto generated - do not modify! if you need to make a change, please see  
the Java Classes in the activemq-openwire-generator module

### 6.445.2 Constructor & Destructor Documentation

**6.445.2.1** `activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::JournalTraceMarshaller  
( ) [inline]`

**6.445.2.2** `virtual activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::~~JournalTraceMarshaller  
( ) [inline, virtual]`

### 6.445.3 Member Function Documentation

**6.445.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::createObject ( )  
const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.445.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::getDataStructureType  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.445.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::looseMarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (  
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.445.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.445.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.445.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.445.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h`

## 6.446 `activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller` Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2300).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMar
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller`:

### Public Member Functions

- **JournalTraceMarshaller** ()
- virtual  $\sim$ **JournalTraceMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.446.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2300). NOTE!  
This file is auto generated - do not modify! if you need to make a change, please see  
the Java Classes in the activemq-openwire-generator module

### 6.446.2 Constructor & Destructor Documentation

**6.446.2.1** `activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::JournalTraceMarshaller  
( ) [inline]`

**6.446.2.2** `virtual activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::~~JournalTraceMarshaller  
( ) [inline, virtual]`

### 6.446.3 Member Function Documentation

**6.446.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::createObject ( )  
const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.446.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::getDataStructureType  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.446.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::looseMarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (  
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

```
6.446.3.4  virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

```
6.446.3.5  virtual int activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.



## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.446.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.446.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshaller.h`

## 6.447 `activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2304).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMar
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller`:

### Public Member Functions

- **JournalTraceMarshaller** ()
- virtual  $\sim$ **JournalTraceMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.447.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2304). NOTE!  
This file is auto generated - do not modify! if you need to make a change, please see  
the Java Classes in the activemq-openwire-generator module

### 6.447.2 Constructor & Destructor Documentation

**6.447.2.1** `activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::JournalTraceMarshaller  
( ) [inline]`

**6.447.2.2** `virtual activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::~~JournalTraceMarshaller  
( ) [inline, virtual]`

### 6.447.3 Member Function Documentation

**6.447.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::createObject ( )  
const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.447.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::getDataStructureType  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.447.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::looseMarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (  
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.447.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.447.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.447.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.447.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshall/v5/JournalTraceMarshaller.h`

## 6.448 `activemq::commands::JournalTransaction` Class Reference

```
#include <src/main/activemq/commands/JournalTransaction.h>
```

Inheritance diagram for `activemq::commands::JournalTransaction`:

### Public Member Functions

- **`JournalTransaction ()`**
- virtual **`~JournalTransaction ()`**
- virtual unsigned char **`getDataStructureType ()`** const  
*Get the unique identifier that this object and its own Marshaller share.*
- virtual **`JournalTransaction * cloneDataStructure ()`** const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **`copyDataStructure (const DataStructure *src)`**  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **`toString ()`** const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **`equals (const DataStructure *value)`** const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const **`Pointer< TransactionId > & getTransactionId ()`** const
- virtual **`Pointer< TransactionId > & getTransactionId ()`**
- virtual void **`setTransactionId (const Pointer< TransactionId > &transactionId)`**
- virtual unsigned char **`getType ()`** const
- virtual void **`setType (unsigned char type)`**
- virtual bool **`getWasPrepared ()`** const
- virtual void **`setWasPrepared (bool wasPrepared)`**

### Static Public Attributes

- static const unsigned char **`ID_JOURNALTRANSACTION = 54`**

### Protected Attributes

- **Pointer**< **TransactionId** > **transactionId**
- unsigned char **type**
- bool **wasPrepared**

### 6.448.1 Constructor & Destructor Documentation

6.448.1.1 **activemq::commands::JournalTransaction::JournalTransaction** ( )

6.448.1.2 **virtual activemq::commands::JournalTransaction::~~JournalTransaction** ( )  
[virtual]

### 6.448.2 Member Function Documentation

6.448.2.1 **virtual JournalTransaction\* activemq::commands::JournalTransaction::cloneDataStructure** ( )  
**const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

6.448.2.2 **virtual void activemq::commands::JournalTransaction::copyDataStructure** ( **const DataStructure \* src** ) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1715).

6.448.2.3 **virtual bool activemq::commands::JournalTransaction::equals** ( **const DataStructure \* value** ) **const** [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns**

true if `DataSet`'s are Equal.

Implements `activemq::commands::DataSet` (p. 1716).

**6.448.2.4** `virtual unsigned char activemq::commands::JournalTransaction::getDataSetType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns**

new `DataSet` (p. 1713) type copy.

Implements `activemq::commands::DataSet` (p. 1717).

**6.448.2.5** `virtual const Pointer<TransactionId>& activemq::commands::JournalTransaction::getTransactionId ( ) const [virtual]`

**6.448.2.6** `virtual Pointer<TransactionId>& activemq::commands::JournalTransaction::getTransactionId ( ) [virtual]`

**6.448.2.7** `virtual unsigned char activemq::commands::JournalTransaction::getType ( ) const [virtual]`

**6.448.2.8** `virtual bool activemq::commands::JournalTransaction::getWasPrepared ( ) const [virtual]`

**6.448.2.9** `virtual void activemq::commands::JournalTransaction::setTransactionId ( const Pointer< TransactionId > & transactionId ) [virtual]`

**6.448.2.10** `virtual void activemq::commands::JournalTransaction::setType ( unsigned char type ) [virtual]`

**6.448.2.11** `virtual void activemq::commands::JournalTransaction::setWasPrepared ( bool wasPrepared ) [virtual]`

**6.448.2.12** `virtual std::string activemq::commands::JournalTransaction::toString ( ) const [virtual]`

Returns a string containing the information for this `DataSet` (p. 1713) such as its type and value of its elements.

**Returns**

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataSet` (p. 841).



## 6.449

**activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller**

**Class Reference**

**2313**

### 6.448.3 Field Documentation

---

6.448.3.1 **const unsigned char activemq::commands::JournalTransaction::ID\_ - JOURNALTRANSACTION = 54** [static]

6.448.3.2 **Pointer<TransactionId> activemq::commands::JournalTransaction::transactionId** [protected]

6.448.3.3 **unsigned char activemq::commands::JournalTransaction::type** [protected]

6.448.3.4 **bool activemq::commands::JournalTransaction::wasPrepared** [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**JournalTransaction.h**

## 6.449 **activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2311).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/JournalTransactionMarsha
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller**:

### Public Member Functions

- **JournalTransactionMarshaller ()**
- virtual **~JournalTransactionMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.449.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2311).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.449.2 Constructor & Destructor Documentation

6.449.2.1 **activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::JournalTransactionMarshaller**  
( ) [**inline**]

6.449.2.2 **virtual activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::~~JournalTransactionMarshaller**  
( ) [**inline**, **virtual**]

### 6.449.3 Member Function Documentation

6.449.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::createObject**  
( ) **const** [**virtual**]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.449

activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller

Class Reference

2315

6.449.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaller.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.449.3.3 virtual void activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

6.449.3.4 virtual void activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

```
6.449.3.5 virtual int activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.449.3.6 virtual void activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

6.450

activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller

Class Reference

2317

6.449.3.7 virtual void activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::tightUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream  
\* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**JournalTransactionMarshaller.h**

## 6.450 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2315).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalTransactionMarsha
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller:

#### Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.450.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2315).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.450.2 Constructor & Destructor Documentation

6.450.2.1 **activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::JournalTransactionMarshaller**  
( ) [inline]

6.450.2.2 **virtual activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::~~JournalTransactionMarshaller**  
( ) [inline, virtual]

### 6.450.3 Member Function Documentation

6.450.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

6.450

**activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller**

**Class Reference**

**2319**

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.450.3.2** virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.450.3.3** virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::looseMarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.450.3.4** virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::looseUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

```
6.450.3.5  virtual int activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.450.3.6  virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



6.451

**activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller**

**Class Reference**

2321

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

6.450.3.7 virtual void **activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::tightUnmarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream**  
\* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**JournalTransactionMarshaller.h**

## 6.451 **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2319).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarsha
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller**:

#### Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual ~**JournalTransactionMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

#### 6.451.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2319).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

#### 6.451.2 Constructor & Destructor Documentation

6.451.2.1 **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::JournalTransactionMarshaller**  
( ) [inline]

6.451.2.2 **virtual activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::~~JournalTransactionMarshaller**  
( ) [inline, virtual]

#### 6.451.3 Member Function Documentation

6.451.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

6.451

**activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller**

**Class Reference**

**2323**

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.451.3.2** virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.451.3.3** virtual void activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::looseMarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.451.3.4** virtual void activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::looseUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

```
6.451.3.5  virtual int activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.451.3.6  virtual void activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

6.452

**activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller**

**Class Reference**

2325

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

6.451.3.7 virtual void **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**JournalTransactionMarshaller.h**

## 6.452 **activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2323).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalTransactionMarsha
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller**:

#### Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual ~**JournalTransactionMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.452.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2323).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.452.2 Constructor & Destructor Documentation

6.452.2.1 **activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::JournalTransactionMarshaller**  
( ) [inline]

6.452.2.2 **virtual activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::~~JournalTransactionMarshaller**  
( ) [inline, virtual]

### 6.452.3 Member Function Documentation

6.452.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

6.452

**activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller**

**Class Reference**

2327

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.452.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.452.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.452.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

```
6.452.3.5  virtual int activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.452.3.6  virtual void activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



6.453

**activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller**

**Class Reference**

2329

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

6.452.3.7 virtual void **activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::tightUnmarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream**  
\* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**JournalTransactionMarshaller.h**

## 6.453 **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2327).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarsha
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller**:

#### Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual ~**JournalTransactionMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.453.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2327).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.453.2 Constructor & Destructor Documentation

6.453.2.1 **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::JournalTransactionMarshaller**  
( ) [inline]

6.453.2.2 **virtual activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::~~JournalTransactionMarshaller**  
( ) [inline, virtual]

### 6.453.3 Member Function Documentation

6.453.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

6.453

**activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller**

**Class Reference**

**2331**

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.453.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.453.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.453.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

```
6.453.3.5  virtual int activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.453.3.6  virtual void activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.454

**activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller**

**Class Reference**

**2333**

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

6.453.3.7 virtual void **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**JournalTransactionMarshaller.h**

## 6.454 **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2331).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarsha
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller**:

### Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual ~**JournalTransactionMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

#### 6.454.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2331).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

#### 6.454.2 Constructor & Destructor Documentation

6.454.2.1 **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::JournalTransactionMarshaller**  
( ) [inline]

6.454.2.2 **virtual activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::~~JournalTransactionMarshaller**  
( ) [inline, virtual]

#### 6.454.3 Member Function Documentation

6.454.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

6.454

**activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller**

**Class Reference**

**2335**

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.454.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.454.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.454.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

```
6.454.3.5  virtual int activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.454.3.6  virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.454.3.7 virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
  * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**JournalTransactionMarshaller.h**

## 6.455 activemq::commands::KeepAliveInfo Class Reference

```
#include <src/main/activemq/commands/KeepAliveInfo.h>
```

Inheritance diagram for **activemq::commands::KeepAliveInfo**:

#### Public Member Functions

- **KeepAliveInfo** ()
- virtual **~KeepAliveInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **KeepAliveInfo \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)

*Copy the contents of the passed object into this object's members, overwriting any existing data.*

- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual bool **isKeepAliveInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

## Static Public Attributes

- static const unsigned char **ID\_KEEPAALIVEINFO** = 10

## 6.455.1 Constructor & Destructor Documentation

6.455.1.1 **activemq::commands::KeepAliveInfo::KeepAliveInfo** ( )

6.455.1.2 **virtual activemq::commands::KeepAliveInfo::~KeepAliveInfo** ( ) [virtual]

## 6.455.2 Member Function Documentation

6.455.2.1 **virtual KeepAliveInfo\*** **activemq::commands::KeepAliveInfo::cloneDataStructure** ( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

## Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

6.455.2.2 **virtual void** **activemq::commands::KeepAliveInfo::copyDataStructure** ( const **DataStructure** \*src ) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

**Parameters**

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

**6.455.2.3** `virtual bool activemq::commands::KeepAliveInfo::equals ( const DataStructure *  
value ) const` [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

**6.455.2.4** `virtual unsigned char activemq::commands::KeepAliveInfo::getDataStructureType ( )  
const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns**

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

**6.455.2.5** `virtual bool activemq::commands::KeepAliveInfo::isKeepAliveInfo ( ) const`  
[inline, virtual]

**Returns**

an answer of true to the **isKeepAliveInfo()** (p. 2337) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 768).

**6.455.2.6** `virtual std::string activemq::commands::KeepAliveInfo::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

**Returns**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

```
6.455.2.7 virtual Pointer<Command> activemq::commands::KeepAliveInfo::visit
( activemq::state::CommandVisitor * visitor ) throw (
exceptions::ActiveMQException ) [virtual]
```

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

### Returns

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

## 6.455.3 Field Documentation

```
6.455.3.1 const unsigned char activemq::commands::KeepAliveInfo::ID_-
KEEPALIVEINFO = 10 [static]
```

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**KeepAliveInfo.h**

## 6.456 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2338).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/KeepAliveInfoMa
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller**:

### Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.456.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2338). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.456.2 Constructor & Destructor Documentation

**6.456.2.1** **activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::KeepAliveInfoMarshaller**  
( ) [inline]

**6.456.2.2** **virtual activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::~~KeepAliveInfoMarshaller**  
( ) [inline, virtual]

### 6.456.3 Member Function Documentation

**6.456.3.1** **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.456.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.456.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 801).

**6.456.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 802).

**6.456.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 803).

**6.456.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 804).

```
6.456.3.7 virtual void activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 806).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**KeepAliveInfoMarshaller.h**

## 6.457 **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2342).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMa
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller**:



## Public Member Functions

- **KeepAliveInfoMarshaller ()**
- virtual **~KeepAliveInfoMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.457.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2342). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.457.2 Constructor & Destructor Documentation

6.457.2.1 `activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::KeepAliveInfoMarshaller ( )` `[inline]`

6.457.2.2 `virtual activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::~~KeepAliveInfoMarshaller ( )` `[inline, virtual]`

## 6.457.3 Member Function Documentation

6.457.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::createObject ( )`  
`const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.457.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::getDataStructureType ( )` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.457.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut )` `throw ( decaf::io::IOException )` `[virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 808).

**6.457.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 809).

**6.457.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 810).

```
6.457.3.6  virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 811).

```
6.457.3.7  virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 813).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h

## 6.458 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2347).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller:

### Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.458.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2347). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.458.2 Constructor & Destructor Documentation

6.458.2.1 `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::KeepAliveInfoMarshaller ( ) [inline]`

6.458.2.2 `virtual activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::~~KeepAliveInfoMarshaller ( ) [inline, virtual]`

### 6.458.3 Member Function Documentation

6.458.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.458.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.458.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 772).

**6.458.3.4** virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::looseUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 774).

**6.458.3.5** virtual int activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::tightMarshal1 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 775).

```
6.458.3.6 virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 776).

```
6.458.3.7 virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.



## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h`

## 6.459 **activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2351).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller**:

## Public Member Functions

- **KeepAliveInfoMarshaller ()**
- virtual **~KeepAliveInfoMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.459.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2351). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.459.2 Constructor & Destructor Documentation

- 6.459.2.1 **activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::KeepAliveInfoMarshaller**  
 ( ) [inline]
- 6.459.2.2 **virtual activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::~~KeepAliveInfoMarshaller**  
 ( ) [inline, virtual]

### 6.459.3 Member Function Documentation

- 6.459.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::createObject** ( )  
 const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

- 6.459.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

## 6.459 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller Class Reference 2355

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.459.3.3** virtual void **activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::looseMarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 780).

**6.459.3.4** virtual void **activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 781).

```

6.459.3.5  virtual int activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 782).

```

6.459.3.6  virtual void activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 783).

## 6.460 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller

### Class Reference

2357

```
6.459.3.7 virtual void activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**KeepAliveInfoMarshaller.h**

## 6.460 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2355).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller**:

#### Public Member Functions

- **KeepAliveInfoMarshaller ()**
- virtual **~KeepAliveInfoMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.460.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2355). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.460.2 Constructor & Destructor Documentation

6.460.2.1 **activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::KeepAliveInfoMarshaller** ( ) [inline]

6.460.2.2 **virtual activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::~~KeepAliveInfoMarshaller** ( ) [inline, virtual]

### 6.460.3 Member Function Documentation

6.460.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::createObject** ( ) **const** [virtual]

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.460.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

## Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.460.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 794).

**6.460.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 795).

```
6.460.3.5  virtual int activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 796).

```
6.460.3.6  virtual void activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled



## 6.461 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller

### Class Reference

2361

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 797).

**6.460.3.7** virtual void activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h

## 6.461 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2359).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller:

## Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.461.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2359). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.461.2 Constructor & Destructor Documentation

6.461.2.1 `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::KeepAliveInfoMarshaller ( ) [inline]`

6.461.2.2 `virtual activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::~~KeepAliveInfoMarshaller ( ) [inline, virtual]`

## 6.461.3 Member Function Documentation

6.461.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.461.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.461.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 787).

**6.461.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 788).

**6.461.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 789).

**6.461.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 790).

**6.461.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 792).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h`

## 6.462 decaf::security::Key Class Reference

The **Key** (p. 2364) interface is the top-level interface for all keys.

```
#include <src/main/decaf/security/Key.h>
```

Inheritance diagram for `decaf::security::Key`:

### Public Member Functions

- virtual `~Key ()`
- virtual `std::string getAlgorithm () const =0`  
*Returns the standard algorithm name for this key.*
- virtual `void getEncoded (std::vector< unsigned char > &output) const =0`  
*Provides the key in its primary encoding format, or nothing if this key does not support encoding.*
- virtual `std::string getFormat () const =0`  
*Returns the name of the primary encoding format of this key, or an empty string if this key does not support encoding.*

### 6.462.1 Detailed Description

The **Key** (p. 2364) interface is the top-level interface for all keys. It defines the functionality shared by all key objects. All keys have three characteristics:

#### An Algorithm

This is the key algorithm for that key. The key algorithm is usually an encryption or asymmetric operation algorithm (such as DSA or RSA), which will work with those algorithms and with related algorithms (such as MD5 with RSA, SHA-1 with RSA, Raw DSA, etc.) The name of the algorithm of a key is obtained using the `getAlgorithm` method.

#### An Encoded Form

This is an external encoded form for the key used when a standard representation of the key is needed outside the application, as when transmitting the key to some other party. The key is encoded according to a standard format (such as X.509 SubjectPublicKeyInfo or PKCS#8), and is returned using the `getEncoded` method. Note: The syntax of the ASN.1 type SubjectPublicKeyInfo is defined as follows:

```
SubjectPublicKeyInfo ::= SEQUENCE {
```

```
algorithm AlgorithmIdentifier,
subjectPublicKey BIT STRING }
AlgorithmIdentifier ::= SEQUENCE {
algorithm OBJECT IDENTIFIER,
parameters ANY DEFINED BY algorithm OPTIONAL }
```

For more information, see RFC 2459: Internet X.509 Public **Key** (p. 2364) Infrastructure Certificate and CRL Profile.

A Format

This is the name of the format of the encoded key. It is returned by the `getFormat` method.

## 6.462.2 Constructor & Destructor Documentation

**6.462.2.1** `virtual decaf::security::Key::~Key ( ) [inline, virtual]`

## 6.462.3 Member Function Documentation

**6.462.3.1** `virtual std::string decaf::security::Key::getAlgorithm ( ) const [pure virtual]`

Returns the standard algorithm name for this key.

For example, "DSA" would indicate that this key is a DSA key.

### Returns

the name of the algorithm associated with this key.

**6.462.3.2** `virtual void decaf::security::Key::getEncoded ( std::vector< unsigned char > & output ) const [pure virtual]`

Provides the key in its primary encoding format, or nothing if this key does not support encoding.

### Parameters

<i>output</i>	Receives the encoded key, or nothing if the key does not support encoding.
---------------	--

**6.462.3.3** `virtual std::string decaf::security::Key::getFormat ( ) const [pure virtual]`

Returns the name of the primary encoding format of this key, or an empty string if this key does not support encoding.

The primary encoding format is named in terms of the appropriate ASN.1 data format,

if an ASN.1 specification for this key exists. For example, the name of the ASN.1 data format for public keys is SubjectPublicKeyInfo, as defined by the X.509 standard; in this case, the returned format is "X.509". Similarly, the name of the ASN.1 data format for private keys is PrivateKeyInfo, as defined by the PKCS #8 standard; in this case, the returned format is "PKCS#8".

### Returns

the primary encoding format of the key.

The documentation for this class was generated from the following file:

- src/main/decaf/security/Key.h

## 6.463 decaf::security::KeyException Class Reference

```
#include <src/main/decaf/security/KeyException.h>
```

Inheritance diagram for decaf::security::KeyException:

### Public Member Functions

- **KeyException** () throw ()  
*Default Constructor.*
- **KeyException** (const decaf::lang::Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **KeyException** (const KeyException &ex) throw ()  
*Copy Constructor.*
- **KeyException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **KeyException** (const std::exception \*cause) throw ()  
*Constructor.*
- **KeyException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **KeyException** \* clone () const  
*Clones this exception.*
- virtual ~**KeyException** () throw ()



### 6.463.1 Constructor & Destructor Documentation

**6.463.1.1** decaf::security::KeyException::KeyException ( ) throw () [inline]

Default Constructor.

**6.463.1.2** decaf::security::KeyException::KeyException ( const decaf::lang::Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.463.1.3** decaf::security::KeyException::KeyException ( const KeyException & ex ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.463.1.4** decaf::security::KeyException::KeyException ( const char \* file, const int lineNumber, const std::exception \* cause, const char \* msg, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.463.1.5** decaf::security::KeyException::KeyException ( const std::exception \* cause ) throw () [inline]

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.463.1.6** `decaf::security::KeyException::KeyException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.  
Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
...	list of primitives that are formatted into the message

**6.463.1.7** `virtual decaf::security::KeyException::~~KeyException ( ) throw ()` `[inline, virtual]`

### 6.463.2 Member Function Documentation

**6.463.2.1** `virtual KeyException* decaf::security::KeyException::clone ( ) const` `[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

A deep copy of this exception.

Reimplemented from `decaf::security::GeneralSecurityException` (p. 2037).

Reimplemented in `decaf::security::InvalidKeyException` (p. 2203), and `decaf::security::KeyManagementException` (p. 2371).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/KeyException.h`

## 6.464 decaf::security::KeyManagementException Class Reference

```
#include <src/main/decaf/security/KeyManagementException.h>
```

Inheritance diagram for `decaf::security::KeyManagementException`:

## Public Member Functions

- **KeyManagementException** () throw ()

*Default Constructor.*

- **KeyManagementException** (const Exception &ex) throw ()

*Conversion Constructor from some other Exception.*

- **KeyManagementException** (const **KeyManagementException** &ex) throw ()

*Copy Constructor.*

- **KeyManagementException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()

*Constructor - Initializes the file name and line number where this message occurred.*

- **KeyManagementException** (const std::exception \*cause) throw ()

*Constructor.*

- **KeyManagementException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()

*Constructor - Initializes the file name and line number where this message occurred.*

- virtual **KeyManagementException** \* clone () const

*Clones this exception.*

- virtual ~**KeyManagementException** () throw ()

### 6.464.1 Constructor & Destructor Documentation

**6.464.1.1** decaf::security::KeyManagementException::KeyManagementException ( ) throw ()  
[inline]

Default Constructor.

**6.464.1.2** decaf::security::KeyManagementException::KeyManagementException ( const  
Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.464.1.3** `decaf::security::KeyManagementException::KeyManagementException ( const KeyManagementException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.464.1.4** `decaf::security::KeyManagementException::KeyManagementException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.464.1.5** `decaf::security::KeyManagementException::KeyManagementException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.464.1.6** `decaf::security::KeyManagementException::KeyManagementException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
<i>...</i>	list of primitives that are formatted into the message

6.464.1.7 virtual `decaf::security::KeyManagementException::~~KeyManagementException ( )`  
`throw () [inline, virtual]`

## 6.464.2 Member Function Documentation

6.464.2.1 virtual `KeyManagementException* decaf::security::KeyManagementException::clone ( ) const`  
`[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

A deep copy of this exception.

Reimplemented from `decaf::security::KeyException` (p. 2368).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/KeyManagementException.h`

## 6.465 activemq::commands::LastPartialCommand Class Reference

```
#include <src/main/activemq/commands/LastPartialCommand.h>
```

Inheritance diagram for `activemq::commands::LastPartialCommand`:

### Public Member Functions

- `LastPartialCommand ()`
- virtual `~LastPartialCommand ()`
- virtual unsigned char `getDataStructureType () const`  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual `LastPartialCommand * cloneDataStructure () const`  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void `copyDataStructure (const DataStructure *src)`  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string `toString () const`

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

- virtual bool **equals** (const **DataStructure** \*value) const

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

## Static Public Attributes

- static const unsigned char **ID\_LASTPARTIALCOMMAND** = 61

## 6.465.1 Constructor & Destructor Documentation

6.465.1.1 **activemq::commands::LastPartialCommand::LastPartialCommand ( )**

6.465.1.2 **virtual activemq::commands::LastPartialCommand::~~LastPartialCommand ( )**  
[virtual]

## 6.465.2 Member Function Documentation

6.465.2.1 **virtual LastPartialCommand\* activemq::commands::LastPartialCommand::cloneDataStructure ( )const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Reimplemented from **activemq::commands::PartialCommand** (p. 3003).

6.465.2.2 **virtual void activemq::commands::LastPartialCommand::copyDataStructure ( const DataStructure \* src )** [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::PartialCommand** (p. 3003).

**6.465.2.3** `virtual bool activemq::commands::LastPartialCommand::equals ( const  
DataStructure * value ) const` [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::PartialCommand** (p. 3004).

**6.465.2.4** `virtual unsigned char activemq::commands::LastPartialCommand::getDataStructureType  
( ) const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1713) type copy.

Reimplemented from **activemq::commands::PartialCommand** (p. 3004).

**6.465.2.5** `virtual std::string activemq::commands::LastPartialCommand::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::PartialCommand** (p. 3005).

### 6.465.3 Field Documentation

**6.465.3.1** `const unsigned char activemq::commands::LastPartialCommand::ID_-  
LASTPARTIALCOMMAND = 61` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/LastPartialCommand.h`

## 6.466 activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2374).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/LastPartialComm
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller:

### Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*



Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2374).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

## 6.466.2 Constructor & Destructor Documentation

6.466.2.1 **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::LastPartialCommandMarshaller**  
 ( ) [inline]

6.466.2.2 **virtual activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::~~LastPartialCommandMarshaller**  
 ( ) [inline, virtual]

## 6.466.3 Member Function Documentation

6.466.3.1 **virtual commands::DataStructure\* ac-**  
**tivemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller**  
 (p. 3007).

6.466.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller**  
 (p. 3007).

6.466.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::looseMarshal**  
 ( **OpenWireFormat \* wireFormat**, **commands::DataStructure**  
 \* **dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller** (p. 3007).

**6.466.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller** (p. 3008).

**6.466.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**6.466 ac-**  
**ativemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller**  
**Class Reference** **2379**  
**Returns**

---

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller** (p. 3008).

**6.466.3.6** virtual void ativemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::tightMarshal2  
( OpenWireFormat \* *wireFormat*, commands::DataStructure  
\* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller** (p. 3009).

**6.466.3.7** virtual void ativemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::tightUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream  
\* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller** (p. 3009).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**LastPartialCommandMarshaller.h**

## 6.467 **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2378).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/LastPartialComm
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller**:

**Public Member Functions**

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

## 6.467 ac-

**ativemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller**

### Class Reference

2381

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.467.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2378).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.467.2 Constructor & Destructor Documentation

6.467.2.1 **ativemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::LastPartialCommandMarshaller**  
( ) [inline]

6.467.2.2 **virtual ativemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::~~LastPartialCommandMarshaller**  
( ) [inline, virtual]

### 6.467.3 Member Function Documentation

6.467.3.1 **virtual commands::DataStructure\* ativemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **ativemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 3020).

6.467.3.2 **virtual unsigned char ativemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 3020).

**6.467.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 3021).

**6.467.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 3021).

6.467 ac-

ativemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller

Class Reference

2383

```
6.467.3.5 virtual int activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 3022).

```
6.467.3.6 virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 3022).

6.467.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 3023).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/LastPartialCommandMarshaller.h`

## 6.468 `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for `LastPartialCommandMarshaller` (p. 2382).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/LastPartialComm
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller`:

#### Public Member Functions

- `LastPartialCommandMarshaller ()`
- `virtual ~LastPartialCommandMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*



## 6.468 ac-

**ativemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller**

### Class Reference

2385

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

## 6.468.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2382).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.468.2 Constructor & Destructor Documentation

6.468.2.1 **ativemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::LastPartialCommandMarshaller**  
( ) [inline]

6.468.2.2 **virtual ativemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::~~LastPartialCommandMarshaller**  
( ) [inline, virtual]

## 6.468.3 Member Function Documentation

6.468.3.1 **virtual commands::DataStructure\* ativemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 3011).

**6.468.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 3011).

**6.468.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 3012).

**6.468.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**6.468 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller**  
**Class Reference** **2387**  
**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 3012).

**6.468.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 3013).

**6.468.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 3013).

**6.468.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 3014).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/LastPartialCommandMarshaller.h`

## 6.469 **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2386).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/LastPartialComm
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller**:

## Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.469.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2386).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.469.2 Constructor & Destructor Documentation

6.469.2.1 **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::LastPartialCommandMarshaller**  
( ) [inline]

6.469.2.2 **virtual activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::~~LastPartialCommandMarshaller**  
( ) [inline, virtual]

## 6.469.3 Member Function Documentation

6.469.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 3016).

6.469.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 3016).

6.469.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::looseMarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**6.469 ac-**  
**ativemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller**  
**Class Reference** **2391**  
**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 3016).

**6.469.3.4** `virtual void ativemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 3017).

**6.469.3.5** `virtual int ativemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 3017).

```
6.469.3.6 virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 3018).

```
6.469.3.7 virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 3018).

The documentation for this class was generated from the following file:



6.470 ac-

ativemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller

Class Reference

2393

- src/main/activemq/wireformat/openwire/marshal/v5/**LastPartialCommandMarshaller.h**

## 6.470 activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2391).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/LastPartialCommandMarsha
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller:

### Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.470.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2391).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.470.2 Constructor & Destructor Documentation

6.470.2.1 **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::LastPartialCommandMarshaller**  
 ( ) [inline]

6.470.2.2 **virtual activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::~~LastPartialCommandMarshaller**  
 ( ) [inline, virtual]

### 6.470.3 Member Function Documentation

6.470.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller**  
 (p. 3025).

6.470.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller**  
 (p. 3025).

## 6.470 ac-

activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller

### Class Reference

2395

```
6.470.3.3 virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::looseMarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (   
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller** (p. 3025).

```
6.470.3.4 virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::looseUnmarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (   
decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller** (p. 3026).

```
6.470.3.5 virtual int activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::tightMarshal1  
( OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )  
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller** (p. 3026).

```
6.470.3.6 virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller** (p. 3027).

```
6.470.3.7 virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

## 6.471 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller

### Class Reference 2397

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller** (p. 3027).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**LastPartialCommandMarshaller.h**

## 6.471 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller

### Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2395).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/LastPartialCommandMarsha
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller**:

### Public Member Functions

- **LastPartialCommandMarshaller ()**
- virtual **~LastPartialCommandMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.471.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2395).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.471.2 Constructor & Destructor Documentation

6.471.2.1 **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::LastPartialCommandMarshaller**  
( ) [inline]

6.471.2.2 **virtual activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::~~LastPartialCommandMarshaller**  
( ) [inline, virtual]

### 6.471.3 Member Function Documentation

6.471.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 3029).

6.471 ac-

ativemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller

Class Reference

2399

6.471.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 3029).

6.471.3.3 virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 3030).

6.471.3.4 virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 3030).

```
6.471.3.5  virtual int activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 3031).

```
6.471.3.6  virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 3031).

**6.471.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 3032).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/LastPartialCommandMarshaller.h`

## 6.472 decaf::util::comparators::Less< E > Class Template Reference

Simple **Less** (p. 2399) **Comparator** (p. 1251) that compares to elements to determine if the first is less than the second.

```
#include <src/main/decaf/util/comparators/Less.h>
```

Inheritance diagram for `decaf::util::comparators::Less< E >`:

**Public Member Functions**

- `Less ()`

- virtual `~Less()`
- virtual `bool operator()(const E &left, const E &right) const`

*Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 1251) to be passed to an **STL Map** (p. 2538) for use as the sorting criteria.*

- virtual `int compare(const E &o1, const E &o2) const`  
*Compares its two arguments for order.*

### 6.472.1 Detailed Description

```
template<typename E> class decaf::util::comparators::Less< E >
```

Simple **Less** (p. 2399) **Comparator** (p. 1251) that compares to elements to determine if the first is less than the second. This can be used in **Collection** (p. 1216) classes to sort elements according to their natural ordering. By design the **Comparator**'s `compare` function return more information about comparison than the STL binary function's boolean compare operator. In this case the `compare` method will return

Since

1.0

### 6.472.2 Constructor & Destructor Documentation

6.472.2.1 `template<typename E> decaf::util::comparators::Less< E >::Less( )`  
[inline]

6.472.2.2 `template<typename E> virtual decaf::util::comparators::Less< E >::~Less( )` [inline, virtual]

### 6.472.3 Member Function Documentation

6.472.3.1 `template<typename E> virtual int decaf::util::comparators::Less< E >::compare( const E & o1, const E & o2 ) const` [inline, virtual]

Compares its two arguments for order.

Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

The implementor must ensure that `sgn( compare(x, y) ) == -sgn(compare(y, x) )` for all `x` and `y`. (This implies that `compare(x, y)` must throw an exception if and only if `compare(y, x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `((compare(x, y)>0) && (compare(y, z)>0))` implies `compare(x, z)>0`.

Finally, the implementer must ensure that `compare(x, y)==0` implies that `sgn(compare(x, z))==sgn(compare(y, z))` for all `z`.

## 6.473 `std::less< decaf::lang::ArrayPointer< T > >` Struct Template Reference

It is generally the case, but not strictly required that `(compare(x, y)==0) == ( x == y )`. Generally speaking, any comparator that violates this condition should clearly indicate this fact. The recommended language is "Note: this comparator imposes orderings that are inconsistent with equals."

### Parameters

<i>o1</i>	- the first object to be compared
<i>o2</i>	- the second object to be compared

### Returns

a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

Implements `decaf::util::Comparator< E >` (p. 1252).

**6.472.3.2** `template<typename E> virtual bool decaf::util::comparators::Less< E >::operator()( const E & left, const E & right ) const` [`inline`, `virtual`]

Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 1251) to be passed to an STL **Map** (p. 2538) for use as the sorting criteria.

### Parameters

<i>left</i>	- the Left hand side operand.
<i>right</i>	- the Right hand side operand.

### Returns

true if the value of left is less than the value of right.

Implements `decaf::util::Comparator< E >` (p. 1252).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/comparators/Less.h`

## 6.473 `std::less< decaf::lang::ArrayPointer< T > >` Struct Template Reference

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

```
#include <src/main/decaf/lang/ArrayPointer.h>
```

Inheritance diagram for `std::less< decaf::lang::ArrayPointer< T > >`:

## Public Member Functions

- **bool operator()** (const **decaf::lang::ArrayPointer**< T > &left, const **decaf::lang::ArrayPointer**< T > &right) const

### 6.473.1 Detailed Description

`template<typename T> struct std::less< decaf::lang::ArrayPointer< T > >`

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

### 6.473.2 Member Function Documentation

**6.473.2.1** `template<typename T> bool std::less< decaf::lang::ArrayPointer< T > >::operator() ( const decaf::lang::ArrayPointer< T > & left, const decaf::lang::ArrayPointer< T > & right ) const [inline]`

References `decaf::lang::ArrayPointer< T, REFCOUNTER >::get()`.

The documentation for this struct was generated from the following file:

- `src/main/decaf/lang/ArrayPointer.h`

## 6.474 `std::less< decaf::lang::Pointer< T > >` Struct Template Reference

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

```
#include <src/main/decaf/lang/Pointer.h>
```

Inheritance diagram for `std::less< decaf::lang::Pointer< T > >`:

## Public Member Functions

- **bool operator()** (const **decaf::lang::Pointer**< T > &left, const **decaf::lang::Pointer**< T > &right) const

### 6.474.1 Detailed Description

```
template<typename T> struct std::less< decaf::lang::Pointer< T > >
```

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

### 6.474.2 Member Function Documentation

6.474.2.1 `template<typename T> bool std::less< decaf::lang::Pointer< T > >::operator() ( const decaf::lang::Pointer< T > & left, const decaf::lang::Pointer< T > & right ) const` [inline]

References decaf::lang::Pointer< T, REFCOUNTER >::get().

The documentation for this struct was generated from the following file:

- src/main/decaf/lang/Pointer.h

## 6.475 decaf::util::logging::Level Class Reference

The **Level** (p. 2403) class defines a set of standard logging levels that can be used to control logging output.

```
#include <src/main/decaf/util/logging/Level.h>
```

Inheritance diagram for decaf::util::logging::Level:

### Public Member Functions

- virtual `~Level ()`
- int `intValue () const`
- std::string `getName () const`
- std::string `toString () const`
- virtual int `compareTo (const Level &value) const`
- virtual bool `equals (const Level &value) const`
- virtual bool `operator== (const Level &value) const`
- virtual bool `operator< (const Level &value) const`

### Static Public Member Functions

- static `Level parse (const std::string &name) throw ( decaf::lang::exceptions::IllegalArgumentException )`  
*Parse a level name string into a Level (p. 2403).*

## Static Public Attributes

- static const **Level INHERIT**

*NULL is a special level that indicates that the **Logger** (p. 2461) should get its **Level** (p. 2403) from its parent **Logger** (p. 2461), the value is initialized as zero.*

- static const **Level OFF**

*OFF is a special level that can be used to turn off logging.*

- static const **Level SEVERE**

*SEVERE is a message level indicating a serious failure.*

- static const **Level WARNING**

*WARNING is a message level indicating a potential problem.*

- static const **Level INFO**

*INFO is a message level for informational messages.*

- static const **Level DEBUG**

*DEBUG is a level for more verbose informative messages.*

- static const **Level CONFIG**

*CONFIG is a message level for static configuration messages.*

- static const **Level FINE**

*FINE is a message level providing tracing information.*

- static const **Level FINER**

*FINER indicates a fairly detailed tracing message.*

- static const **Level FINEST**

*FINEST indicates a highly detailed tracing message.*

- static const **Level ALL**

*ALL indicates that all messages should be logged.*

## Protected Member Functions

- **Level** (const std::string &name, int value)

*Create a named **Level** (p. 2403) with a given integer value.*

### 6.475.1 Detailed Description

The **Level** (p. 2403) class defines a set of standard logging levels that can be used to control logging output. The logging **Level** (p. 2403) objects are ordered and are specified by ordered integers. Enabling logging at a given level also enables logging at all higher levels.

Clients should normally use the predefined **Level** (p. 2403) constants such as **Level.SEVERE** (p. 2408).

The levels in descending order are:

\* SEVERE (highest value) \* WARNING \* INFO \* DEBUG \* CONFIG \* FINE \* FINER \* FINEST (lowest value)

In addition there is a level OFF that can be used to turn off logging, and a level ALL that can be used to enable logging of all messages.

It is possible for third parties to define additional logging levels by subclassing **Level** (p. 2403). In such cases subclasses should take care to chose unique integer level values.

**Since**

1.0

### 6.475.2 Constructor & Destructor Documentation

**6.475.2.1** `decaf::util::logging::Level::Level ( const std::string & name, int value )`  
[protected]

Create a named **Level** (p. 2403) with a given integer value.

#### Parameters

<i>name</i>	Name of the level, e.g. SEVERE
<i>value</i>	Unique integer value of this level, e.g. 100

6.475.2.2 `virtual decaf::util::logging::Level::~~Level ( ) [virtual]`

### 6.475.3 Member Function Documentation

6.475.3.1 `virtual int decaf::util::logging::Level::compareTo ( const Level & value ) const [virtual]`

6.475.3.2 `virtual bool decaf::util::logging::Level::equals ( const Level & value ) const [virtual]`

6.475.3.3 `std::string decaf::util::logging::Level::getName ( ) const [inline]`

#### Returns

the name of this **Level** (p. 2403) instance.

6.475.3.4 `int decaf::util::logging::Level::intValue ( ) const [inline]`

#### Returns

the integer value of this level instance.

6.475.3.5 `virtual bool decaf::util::logging::Level::operator< ( const Level & value ) const [virtual]`

6.475.3.6 `virtual bool decaf::util::logging::Level::operator== ( const Level & value ) const [virtual]`

6.475.3.7 `static Level decaf::util::logging::Level::parse ( const std::string & name ) throw ( decaf::lang::exceptions::IllegalArgumentException ) [static]`

Parse a level name string into a **Level** (p. 2403).

The argument string may consist of either a level name or an integer value.

For example:

\* "SEVERE" \* "1000"

#### Parameters

<i>name</i>	- The name or int value of the desired <b>Level</b> (p. 2403)
-------------	---

#### Returns

the parsed **Level** (p. 2403) value, passing in a level name that is an int value that is not one of the known **Level** (p. 2403) values will result in a new **Level** (p. 2403) that has been initialized with that int value and name as the string form of the int.



**Exceptions**

<i>IllegalArgumentException</i>	if the value is not valid, validity means that the string is either a valid int (between Integer::MIN_VALUE and Integer::MAX_VALUE or is one of the known level names.
---------------------------------	--

**6.475.3.8** `std::string decaf::util::logging::Level::toString ( ) const` `[inline]`

**Returns**

the string value of this **Level** (p. 2403), e.g. "SEVERE".

**6.475.4 Field Documentation**

**6.475.4.1** `const Level decaf::util::logging::Level::ALL` `[static]`

ALL indicates that all messages should be logged.

This level is initialized to Integer::MIN\_VALUE.

**6.475.4.2** `const Level decaf::util::logging::Level::CONFIG` `[static]`

CONFIG is a message level for static configuration messages.

CONFIG messages are intended to provide a variety of static configuration information, to assist in debugging problems that may be associated with particular configurations. For example, CONFIG message might include the CPU type, the System properties, etc. This level is initialized to 600.

**6.475.4.3** `const Level decaf::util::logging::Level::DEBUG` `[static]`

DEBUG is a level for more verbose informative messages.

DEBUG messages are intended to provide a more detailed message intended for use by developers in tracking the behavior of a client. DEBUG messages typically contain more implementation specific information that might not be significant to end users or system admins. This level is initialized to 700.

**6.475.4.4** `const Level decaf::util::logging::Level::FINE` `[static]`

FINE is a message level providing tracing information.

All of FINE, FINER, and FINEST are intended for relatively detailed tracing. The exact meaning of the three levels will vary between subsystems, but in general, FINEST should be used for the most detailed output, FINER for somewhat less detailed output, and FINE for the lowest volume (and most important) messages.

In general the FINE level should be used for information that will be broadly interesting to developers who do not have a specialized interest in the specific subsystem.

FINE messages might include things like minor (recoverable) failures. Issues indicating potential performance problems are also worth logging as FINE. This level is initialized to 500.

#### 6.475.4.5 `const Level decaf::util::logging::Level::FINER` [static]

FINER indicates a fairly detailed tracing message.

By default logging calls for entering, returning, or throwing an exception are traced at this level. This level is initialized to 400.

#### 6.475.4.6 `const Level decaf::util::logging::Level::FINEST` [static]

FINEST indicates a highly detailed tracing message.

This level is initialized to 300.

#### 6.475.4.7 `const Level decaf::util::logging::Level::INFO` [static]

INFO is a message level for informational messages.

Typically INFO messages will be written to the console or its equivalent. So the INFO level should only be used for reasonably significant messages that will make sense to end users and system admins. This level is initialized to 800.

#### 6.475.4.8 `const Level decaf::util::logging::Level::INHERIT` [static]

NULL is a special level that indicates that the **Logger** (p. 2461) should get its **Level** (p. 2403) from its parent **Logger** (p. 2461), the value is initialized as zero.

#### 6.475.4.9 `const Level decaf::util::logging::Level::OFF` [static]

OFF is a special level that can be used to turn off logging.

This level is initialized to `Integer::MAX_VALUE`

#### 6.475.4.10 `const Level decaf::util::logging::Level::SEVERE` [static]

SEVERE is a message level indicating a serious failure.

In general SEVERE messages should describe events that are of considerable importance and which will prevent normal program execution. They should be reasonably intelligible to end users and to system administrators. This level is initialized to 1000.

#### 6.475.4.11 `const Level decaf::util::logging::Level::WARNING` [static]

WARNING is a message level indicating a potential problem.

In general WARNING messages should describe events that will be of interest to end users or system managers, or which indicate potential problems. This level is initialized to 900.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/Level.h

## 6.476 decaf::util::List< E > Class Template Reference

An ordered collection (also known as a sequence).

```
#include <src/main/decaf/util/List.h>
```

Inheritance diagram for decaf::util::List< E >:

### Public Member Functions

- **List** ()
- virtual **~List** ()
- virtual **ListIterator**< E > \* **listIterator** ()=0
- virtual **ListIterator**< E > \* **listIterator** () const=0
- virtual **ListIterator**< E > \* **listIterator** (std::size\_t index)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual **ListIterator**< E > \* **listIterator** (std::size\_t index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual std::size\_t **indexOf** (const E &value)=0 throw ( decaf::lang::exceptions::NoSuchElementException )  
*Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.*
- virtual std::size\_t **lastIndexOf** (const E &value)=0 throw ( decaf::lang::exceptions::NoSuchElementException )  
*Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.*
- virtual E **get** (std::size\_t index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Gets the element contained at position passed.*
- virtual E **set** (std::size\_t index, const E &element)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Replaces the element at the specified position in this list with the specified element.*
- virtual void **add** (std::size\_t index, const E &element)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException decaf::lang::exceptions::IndexOutOfBoundsException )

*Inserts the specified element at the specified position in this list.*

- virtual bool **addAll** (std::size\_t index, const **Collection**< E > &source)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Inserts all of the elements in the specified collection into this list at the specified position (optional operation).*

- virtual E **remove** (std::size\_t index)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Removes the element at the specified position in this list.*

### 6.476.1 Detailed Description

```
template<typename E> class decaf::util::List< E >
```

An ordered collection (also known as a sequence). The user of this interface has precise control over where in the list each element is inserted. The user can access elements by their integer index (position in the list), and search for elements in the list.

Unlike sets, lists typically allow duplicate elements. More formally, lists typically allow pairs of elements e1 and e2 such that e1.equals(e2), and they typically allow multiple null elements if they allow null elements at all. It is not inconceivable that someone might wish to implement a list that prohibits duplicates, by throwing runtime exceptions when the user attempts to insert them, but we expect this usage to be rare.

### 6.476.2 Constructor & Destructor Documentation

6.476.2.1 `template<typename E> decaf::util::List< E >::List ( ) [inline]`

6.476.2.2 `template<typename E> virtual decaf::util::List< E >::~~List ( ) [inline, virtual]`

### 6.476.3 Member Function Documentation

6.476.3.1 `template<typename E> virtual void decaf::util::List< E >::add ( std::size_t index, const E & element ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

**Parameters**

<i>index</i>	- index at which the specified element is to be inserted
<i>element</i>	- element to be inserted

**Exceptions**

<i>IndexOutOfBoundsException</i>	- if the index is greater than size
<i>UnsupportedOperationException</i>	- If the collection is non-modifiable.

Implemented in **decaf::util::StlList< E >** (p. 3701), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3701), **decaf::util::StlList< CompositeTask \* >** (p. 3701), **decaf::util::StlList< URI >** (p. 3701), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3701), **decaf::util::StlList< PrimitiveValueNode >** (p. 3701), **decaf::util::StlList< Pointer< Command > >** (p. 3701), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3701), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3701), **decaf::util::StlList< cms::Destination \* >** (p. 3701), **decaf::util::StlList< cms::Session \* >** (p. 3701), and **decaf::util::StlList< cms::Connection \* >** (p. 3701).

```
6.476.3.2  template<typename E> virtual bool decaf::util::List< E >::addAll
            ( std::size_t index, const Collection< E > & source ) throw (
              decaf::lang::exceptions::UnsupportedOperationException,
              decaf::lang::exceptions::IndexOutOfBoundsException ) [pure
              virtual]
```

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

**Parameters**

<i>index</i>	The index at which to insert the first element from the specified collection
<i>source</i>	The <b>Collection</b> (p. 1216) containing elements to be added to this list

**Returns**

true if this list changed as a result of the call

**Exceptions**

<i>IndexOutOfBoundsException</i>	- if the index is greater than size
<i>UnsupportedOperationException</i>	- If the collection is non-modifiable.

Implemented in `decaf::util::StlList< E >` (p. 3701), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3701), `decaf::util::StlList< CompositeTask * >` (p. 3701), `decaf::util::StlList< URI >` (p. 3701), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3701), `decaf::util::StlList< PrimitiveValueNode >` (p. 3701), `decaf::util::StlList< Pointer< Command > >` (p. 3701), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3701), `decaf::util::StlList< cms::MessageProducer * >` (p. 3701), `decaf::util::StlList< cms::Destination * >` (p. 3701), `decaf::util::StlList< cms::Session * >` (p. 3701), and `decaf::util::StlList< cms::Connection * >` (p. 3701).

**6.476.3.3** `template<typename E> virtual E decaf::util::List< E >::get ( std::size_t index )`  
`const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
`[pure virtual]`

Gets the element contained at position passed.

#### Parameters

<i>index</i>	- position to get
--------------	-------------------

#### Returns

value at index

Implemented in `decaf::util::StlList< E >` (p. 3703), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3703), `decaf::util::StlList< CompositeTask * >` (p. 3703), `decaf::util::StlList< URI >` (p. 3703), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3703), `decaf::util::StlList< PrimitiveValueNode >` (p. 3703), `decaf::util::StlList< Pointer< Command > >` (p. 3703), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3703), `decaf::util::StlList< cms::MessageProducer * >` (p. 3703), `decaf::util::StlList< cms::Destination * >` (p. 3703), `decaf::util::StlList< cms::Session * >` (p. 3703), and `decaf::util::StlList< cms::Connection * >` (p. 3703).

**6.476.3.4** `template<typename E> virtual std::size_t decaf::util::List< E >::indexOf ( const E & value ) throw ( decaf::lang::exceptions::NoSuchElementException )`  
`[pure virtual]`

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the lowest index *i* such that `get(i) == value`, or -1 if there is no such index.

#### Parameters

<i>value</i>	- element to search for
--------------	-------------------------

#### Returns

the index of the first occurrence of the specified element in this list,

**Exceptions**

<i>NoSuchElementException</i>	if value is not in the list
-------------------------------	-----------------------------

Implemented in **decaf::util::StlList< E >** (p. 3703), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3703), **decaf::util::StlList< CompositeTask \* >** (p. 3703), **decaf::util::StlList< URI >** (p. 3703), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3703), **decaf::util::StlList< PrimitiveValueNode >** (p. 3703), **decaf::util::StlList< Pointer< Command > >** (p. 3703), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3703), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3703), **decaf::util::StlList< cms::Destination \* >** (p. 3703), **decaf::util::StlList< cms::Session \* >** (p. 3703), and **decaf::util::StlList< cms::Connection \* >** (p. 3703).

```
6.476.3.5  template<typename E> virtual size_t decaf::util::List< E >::lastIndexOf( const
           E & value ) throw ( decaf::lang::exceptions::NoSuchElementException )
           [pure virtual]
```

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the highest index *i* such that `get(i) == value` or -1 if there is no such index.

**Parameters**

<i>value</i>	- element to search for
--------------	-------------------------

**Returns**

the index of the last occurrence of the specified element in this list.

**Exceptions**

<i>NoSuchElementException</i>	if value is not in the list
-------------------------------	-----------------------------

Implemented in **decaf::util::StlList< E >** (p. 3704), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3704), **decaf::util::StlList< CompositeTask \* >** (p. 3704), **decaf::util::StlList< URI >** (p. 3704), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3704), **decaf::util::StlList< PrimitiveValueNode >** (p. 3704), **decaf::util::StlList< Pointer< Command > >** (p. 3704), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3704), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3704), **decaf::util::StlList< cms::Destination \* >** (p. 3704), **decaf::util::StlList< cms::Session \* >** (p. 3704), and **decaf::util::StlList< cms::Connection \* >** (p. 3704).

```
6.476.3.6  template<typename E> virtual ListIterator<E>* decaf::util::List< E
           >::listIterator ( ) [pure virtual]
```

### Returns

a list iterator over the elements in this list (in proper sequence).

Implemented in **decaf::util::StlList< E >** (p. 3705), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3705), **decaf::util::StlList< CompositeTask \* >** (p. 3705), **decaf::util::StlList< URI >** (p. 3705), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3705), **decaf::util::StlList< PrimitiveValueNode >** (p. 3705), **decaf::util::StlList< Pointer< Command > >** (p. 3705), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3705), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3705), **decaf::util::StlList< cms::Destination \* >** (p. 3705), **decaf::util::StlList< cms::Session \* >** (p. 3705), and **decaf::util::StlList< cms::Connection \* >** (p. 3705).

```
6.476.3.7  template<typename E> virtual ListIterator<E>*
           decaf::util::List< E >::listIterator ( std::size_t index ) const throw (
           decaf::lang::exceptions::IndexOutOfBoundsException ) [pure
           virtual]
```

Implemented in **decaf::util::StlList< E >** (p. 3706), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3706), **decaf::util::StlList< CompositeTask \* >** (p. 3706), **decaf::util::StlList< URI >** (p. 3706), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3706), **decaf::util::StlList< PrimitiveValueNode >** (p. 3706), **decaf::util::StlList< Pointer< Command > >** (p. 3706), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3706), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3706), **decaf::util::StlList< cms::Destination \* >** (p. 3706), **decaf::util::StlList< cms::Session \* >** (p. 3706), and **decaf::util::StlList< cms::Connection \* >** (p. 3706).

```
6.476.3.8  template<typename E> virtual ListIterator<E>* decaf::util::List< E
           >::listIterator ( ) const [pure virtual]
```

Implemented in **decaf::util::StlList< E >** (p. 3706), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3706), **decaf::util::StlList< CompositeTask \* >** (p. 3706), **decaf::util::StlList< URI >** (p. 3706), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3706), **decaf::util::StlList< PrimitiveValueNode >** (p. 3706), **decaf::util::StlList< Pointer< Command > >** (p. 3706), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3706), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3706), **decaf::util::StlList< cms::Destination \* >** (p. 3706), **decaf::util::StlList< cms::Session \* >** (p. 3706), and **decaf::util::StlList< cms::Connection \* >** (p. 3706).



```

6.476.3.9 template<typename E> virtual ListIterator<E>*
    decaf::util::List< E >::listIterator ( std::size_t index ) throw (
    decaf::lang::exceptions::IndexOutOfBoundsException ) [pure
    virtual]

```

**Parameters**

<i>index</i>	index of first element to be returned from the list iterator (by a call to the next method).
--------------	--

**Returns**

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if the index is out of range (index < 0    index > size() (p. 1226))
----------------------------------	--

Implemented in **decaf::util::StlList< E >** (p. 3705), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3705), **decaf::util::StlList< CompositeTask \* >** (p. 3705), **decaf::util::StlList< URI >** (p. 3705), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3705), **decaf::util::StlList< PrimitiveValueNode >** (p. 3705), **decaf::util::StlList< Pointer< Command > >** (p. 3705), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3705), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3705), **decaf::util::StlList< cms::Destination \* >** (p. 3705), **decaf::util::StlList< cms::Session \* >** (p. 3705), and **decaf::util::StlList< cms::Connection \* >** (p. 3705).

```

6.476.3.10 template<typename E> virtual E decaf::util::List< E >::remove ( std::size_t index ) throw ( decaf::lang::exceptions::UnsupportedOperationException,
    decaf::lang::exceptions::IndexOutOfBoundsException ) [pure
    virtual]

```

Removes the element at the specified position in this list.

Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

**Parameters**

<i>index</i>	- the index of the element to be removed
--------------	--

**Returns**

the element previously at the specified position

**Exceptions**

<i>IndexOutOfBoundsException</i>	- if the index is greater than size
----------------------------------	-------------------------------------

<i>UnsupportedOperationException</i>	- If the collection is non-modifiable.
--------------------------------------	--

Implemented in **decaf::util::StlList< E >** (p. 3706), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3706), **decaf::util::StlList< CompositeTask \* >** (p. 3706), **decaf::util::StlList< URI >** (p. 3706), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3706), **decaf::util::StlList< PrimitiveValueNode >** (p. 3706), **decaf::util::StlList< Pointer< Command > >** (p. 3706), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3706), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3706), **decaf::util::StlList< cms::Destination \* >** (p. 3706), **decaf::util::StlList< cms::Session \* >** (p. 3706), and **decaf::util::StlList< cms::Connection \* >** (p. 3706).

```
6.476.3.11  template<typename E> virtual E decaf::util::List< E
>::set ( std::size_t index, const E & element ) throw (
decaf::lang::exceptions::IndexOutOfBoundsException ) [pure
virtual]
```

Replaces the element at the specified position in this list with the specified element.

#### Parameters

<i>index</i>	- index of the element to replace
<i>element</i>	- element to be stored at the specified position

#### Returns

the element previously at the specified position

#### Exceptions

<i>IndexOutOfBoundsException</i>	- if the index is greater than size
----------------------------------	-------------------------------------

Implemented in **decaf::util::StlList< E >** (p. 3707), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3707), **decaf::util::StlList< CompositeTask \* >** (p. 3707), **decaf::util::StlList< URI >** (p. 3707), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3707), **decaf::util::StlList< PrimitiveValueNode >** (p. 3707), **decaf::util::StlList< Pointer< Command > >** (p. 3707), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3707), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3707), **decaf::util::StlList< cms::Destination \* >** (p. 3707), **decaf::util::StlList< cms::Session \* >** (p. 3707), and **decaf::util::StlList< cms::Connection \* >** (p. 3707).

The documentation for this class was generated from the following file:

- src/main/decaf/util/List.h

## 6.477 decaf::util::ListIterator< E > Class Template Reference

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.

```
#include <src/main/decaf/util/ListIterator.h>
```

Inheritance diagram for decaf::util::ListIterator< E >:

### Public Member Functions

- virtual **~ListIterator** ()
- virtual void **add** (const E &e)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException )  
*Inserts the specified element into the list (optional operation).*
- virtual void **set** (const E &e)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Replaces the last element returned by next or previous with the specified element (optional operation).*
- virtual bool **hasPrevious** () const =0  
*Returns true if this list iterator has more elements when traversing the list in the reverse direction.*
- virtual E **previous** ()=0  
*Returns the previous element in the list.*
- virtual int **nextIndex** () const =0  
*Returns the index of the element that would be returned by a subsequent call to next.*
- virtual int **previousIndex** () const =0  
*Returns the index of the element that would be returned by a subsequent call to previous.*

### 6.477.1 Detailed Description

```
template<typename E> class decaf::util::ListIterator< E >
```

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list. Note that the **remove()** (p. 2223) and **set(Object)** methods are not defined in terms of the cursor position; they are defined to operate on the last element returned by a call to **next()** (p. 2223) or **previous()** (p. 2419).

## 6.477.2 Constructor & Destructor Documentation

6.477.2.1 `template<typename E> virtual decaf::util::ListIterator< E>::~~ListIterator  
( ) [inline, virtual]`

## 6.477.3 Member Function Documentation

6.477.3.1 `template<typename E> virtual void decaf::util::ListIterator< E>::add( const  
E & e ) throw ( decaf::lang::exceptions::UnsupportedOperationException,  
decaf::lang::exceptions::IllegalArgumentException ) [pure  
virtual]`

Inserts the specified element into the list (optional operation).

The element is inserted immediately before the next element that would be returned by next, if any, and after the next element that would be returned by previous, if any. (If the list contains no elements, the new element becomes the sole element on the list.) The new element is inserted before the implicit cursor: a subsequent call to next would be unaffected, and a subsequent call to previous would return the new element. (This call increases by one the value that would be returned by a call to nextIndex or previousIndex.)

### Parameters

<i>e</i>	- the element to insert.
----------	--------------------------

### Exceptions

<i>UnsupportedOperationException</i>	- if the add method is not supported by this list iterator.
<i>IllegalArgumentException</i>	- if some aspect of this element prevents it from being added to this list.

6.477.3.2 `template<typename E> virtual bool decaf::util::ListIterator< E  
>::hasPrevious( ) const [pure virtual]`

Returns true if this list iterator has more elements when traversing the list in the reverse direction.

(In other words, returns true if previous would return an element rather than throwing an exception.)

### Returns

true if the list iterator has more elements when traversing the list in the reverse direction.

**6.477.3.3** `template<typename E> virtual int decaf::util::ListIterator< E >::nextIndex ( ) const [pure virtual]`

Returns the index of the element that would be returned by a subsequent call to next.  
(Returns list size if the list iterator is at the end of the list.)

#### Returns

the index of the element that would be returned by a subsequent call to next, or list size if list iterator is at end of list.

**6.477.3.4** `template<typename E> virtual E decaf::util::ListIterator< E >::previous ( ) [pure virtual]`

Returns the previous element in the list.

This method may be called repeatedly to iterate through the list backwards, or inter-mixed with calls to next to go back and forth. (Note that alternating calls to next and previous will return the same element repeatedly.)

#### Returns

the previous element in the list.

#### Exceptions

<i>NoSuchElementException</i>	- if the iteration has no previous element.
-------------------------------	---

**6.477.3.5** `template<typename E> virtual int decaf::util::ListIterator< E >::previousIndex ( ) const [pure virtual]`

Returns the index of the element that would be returned by a subsequent call to previous.

(Returns -1 if the list iterator is at the beginning of the list.)

#### Returns

the index of the element that would be returned by a subsequent call to previous, or -1 if list iterator is at beginning of list.

6.477.3.6 `template<typename E > virtual void decaf::util::ListIterator< E >::set ( const E & e ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException ) [pure virtual]`

Replaces the last element returned by next or previous with the specified element (optional operation).

This call can be made only if neither **ListIterator.remove** (p. 2223) nor **ListIterator.add** (p. 2418) have been called after the last call to next or previous.

#### Parameters

<i>e</i>	- the element with which to replace the last element returned by next or previous.
----------	--

#### Exceptions

<i>UnsupportedOperationException</i>	- if the add method is not supported by this list iterator.
<i>IllegalArgumentException</i>	- if some aspect of this element prevents it from being added to this list.
<i>IllegalStateException</i>	- if neither next nor previous have been called, or remove or add have been called after the last call to next or previous.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/ListIterator.h`

## 6.478 activemq::commands::LocalTransactionId Class Reference

```
#include <src/main/activemq/commands/LocalTransactionId.h>
```

Inheritance diagram for `activemq::commands::LocalTransactionId`:

#### Public Types

- `typedef decaf::lang::PointerComparator< LocalTransactionId > COMPARATOR`

#### Public Member Functions

- `LocalTransactionId ()`
- `LocalTransactionId (const LocalTransactionId &other)`
- `virtual ~LocalTransactionId ()`
- `virtual unsigned char getDataStructureType () const`

*Get the unique identifier that this object and its own Marshaler share.*

- virtual **LocalTransactionId** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual long long **getValue** () const
- virtual void **setValue** (long long value)
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual int **compareTo** (const **LocalTransactionId** &value) const
- virtual bool **equals** (const **LocalTransactionId** &value) const
- virtual bool **operator==** (const **LocalTransactionId** &value) const
- virtual bool **operator<** (const **LocalTransactionId** &value) const
- **LocalTransactionId** & **operator=** (const **LocalTransactionId** &other)

### Static Public Attributes

- static const unsigned char **ID\_LOCALTRANSACTIONID** = 111

### Protected Attributes

- long long value
- **Pointer**< **ConnectionId** > connectionId

## 6.478.1 Member Typedef Documentation

- 6.478.1.1 typedef decaf::lang::PointerComparator<LocalTransactionId>  
activemq::commands::LocalTransactionId::COMPARATOR

Reimplemented from **activemq::commands::TransactionId** (p. 3933).

## 6.478.2 Constructor & Destructor Documentation

6.478.2.1 `activemq::commands::LocalTransactionId::LocalTransactionId ( )`

6.478.2.2 `activemq::commands::LocalTransactionId::LocalTransactionId ( const LocalTransactionId & other )`

6.478.2.3 `virtual activemq::commands::LocalTransactionId::~~LocalTransactionId ( )`  
[virtual]

## 6.478.3 Member Function Documentation

6.478.3.1 `virtual LocalTransactionId* activemq::commands::LocalTransactionId::cloneDataStructure ( )`  
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Reimplemented from `activemq::commands::TransactionId` (p. 3933).

6.478.3.2 `virtual int activemq::commands::LocalTransactionId::compareTo ( const LocalTransactionId & value ) const` [virtual]

6.478.3.3 `virtual void activemq::commands::LocalTransactionId::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from `activemq::commands::TransactionId` (p. 3934).

6.478.3.4 `virtual bool activemq::commands::LocalTransactionId::equals ( const DataStructure * value ) const` [virtual]

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.



**Returns**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::TransactionId** (p. 3934).

**6.478.3.5** `virtual bool activemq::commands::LocalTransactionId::equals ( const LocalTransactionId & value ) const [virtual]`

**6.478.3.6** `virtual const Pointer<ConnectionId>& activemq::commands::LocalTransactionId::getConnectionId ( ) const [virtual]`

**6.478.3.7** `virtual Pointer<ConnectionId>& activemq::commands::LocalTransactionId::getConnectionId ( ) [virtual]`

**6.478.3.8** `virtual unsigned char activemq::commands::LocalTransactionId::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns**

new **DataStructure** (p. 1713) type copy.

Reimplemented from **activemq::commands::TransactionId** (p. 3934).

- 6.478.3.9 `virtual long long activemq::commands::LocalTransactionId::getValue ( ) const`  
[virtual]
- 6.478.3.10 `virtual bool activemq::commands::LocalTransactionId::operator< ( const`  
`LocalTransactionId & value ) const` [virtual]
- 6.478.3.11 `LocalTransactionId& activemq::commands::LocalTransactionId::operator= (`  
`const LocalTransactionId & other )`
- 6.478.3.12 `virtual bool activemq::commands::LocalTransactionId::operator== ( const`  
`LocalTransactionId & value ) const` [virtual]
- 6.478.3.13 `virtual void activemq::commands::LocalTransactionId::setConnectionId ( const`  
`Pointer< ConnectionId > & connectionId )` [virtual]
- 6.478.3.14 `virtual void activemq::commands::LocalTransactionId::setValue ( long long value )`  
[virtual]
- 6.478.3.15 `virtual std::string activemq::commands::LocalTransactionId::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::TransactionId` (p. 3935).

#### 6.478.4 Field Documentation

- 6.478.4.1 `Pointer<ConnectionId> activemq::commands::LocalTransactionId::connectionId`  
[protected]
- 6.478.4.2 `const unsigned char activemq::commands::LocalTransactionId::ID_-`  
`LOCALTRANSACTIONID = 111` [static]
- 6.478.4.3 `long long activemq::commands::LocalTransactionId::value`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/LocalTransactionId.h`

6.479

activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller

Class Reference

6.479 — activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller <sup>2427</sup>

## Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2425).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/LocalTransactionIdMarsha
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller:

## Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.479.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2425).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.479.2 Constructor & Destructor Documentation

6.479.2.1 **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller**  
 ( ) [inline]

6.479.2.2 **virtual activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::~~LocalTransactionIdMarshaller**  
 ( ) [inline, virtual]

### 6.479.3 Member Function Documentation

6.479.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.479.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.479.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::looseMarshal**  
 ( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

## 6.479

**activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller**

### Class Reference

2429

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** (p. 3957).

**6.479.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** (p. 3957).

**6.479.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3958).

```
6.479.3.6 virtual void activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3958).

```
6.479.3.7 virtual void activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

6.480

**activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller**

**Class Reference**

**2431**

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** (p. 3959).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**LocalTransactionIdMarshaller.h**

## 6.480 **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2429).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarsha
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller**:

### Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.480.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2429).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.480.2 Constructor & Destructor Documentation

6.480.2.1 **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller**  
 ( ) [inline]

6.480.2.2 **virtual activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::~~LocalTransactionIdMarshaller**  
 ( ) [inline, virtual]

### 6.480.3 Member Function Documentation

6.480.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.480.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::getDataStructure**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value



6.480

**activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller**

**Class Reference**

**2433**

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.480.3.3** virtual void **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::looseMarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3945).

**6.480.3.4** virtual void **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3945).

```

6.480.3.5  virtual int activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3946).

```

6.480.3.6  virtual void activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3946).

6.481

activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller

Class Reference

2435

6.480.3.7 virtual void activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3947).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**LocalTransactionIdMarshaller.h**

## 6.481 activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2433).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarsha
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller**:

#### Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.481.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2433).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.481.2 Constructor & Destructor Documentation

6.481.2.1 **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller**  
( ) [inline]

6.481.2.2 **virtual activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::~~LocalTransactionIdMarshaller**  
( ) [inline, virtual]

### 6.481.3 Member Function Documentation

6.481.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

6.481

**activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller**

**Class Reference**

**2437**

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.481.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.481.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3949).

**6.481.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3949).

```
6.481.3.5  virtual int activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3950).

```
6.481.3.6  virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

## 6.482

**activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller**

**Class Reference**

**2439**

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3950).

**6.481.3.7** virtual void **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3951).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**LocalTransactionIdMarshaller.h**

## 6.482 **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2437).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/LocalTransactionIdMarsha
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller**:

## Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.482.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2437).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



6.482

activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller

Class Reference

2441

## 6.482.2 Constructor & Destructor Documentation

6.482.2.1 `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller ( ) [inline]`

6.482.2.2 `virtual activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::~~LocalTransactionIdMarshaller ( ) [inline, virtual]`

## 6.482.3 Member Function Documentation

6.482.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.482.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.482.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3937).

**6.482.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3937).

**6.482.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

6.482

activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller

Class Reference

2443

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3938).

**6.482.3.6** virtual void activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::tightMarshal2 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** )  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3938).

**6.482.3.7** virtual void activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3939).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/LocalTransactionIdMarshaller.h`

## 6.483 `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller` Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2442).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller`:

### Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

6.483

**activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller**

**Class Reference**

**2445**

---

*Write a object instance to data output stream.*

### 6.483.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2442).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.483.2 Constructor & Destructor Documentation

6.483.2.1 **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller**  
( ) [inline]

6.483.2.2 **virtual activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::~~LocalTransactionIdMarshaller**  
( ) [inline, virtual]

### 6.483.3 Member Function Documentation

6.483.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.483.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.483.3.3 **virtual void activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3953).

**6.483.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3953).

**6.483.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

6.483

**activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller****Class Reference****2447****Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3954).

**6.483.3.6** virtual void **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::tightMarshal2** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** )  
[virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3954).

**6.483.3.7** virtual void **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3955).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarshaller.h`

## 6.484 **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2446).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller**:

**Public Member Functions**

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*



6.484

**activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller**

**Class Reference**

2449

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.484.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2446).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.484.2 Constructor & Destructor Documentation

6.484.2.1 **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller**  
( ) [inline]

6.484.2.2 **virtual activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::~~LocalTransactionIdMarshaller**  
( ) [inline, virtual]

### 6.484.3 Member Function Documentation

6.484.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.484.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.484.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3941).

**6.484.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3941).

6.484

activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller

Class Reference

2451

```
6.484.3.5 virtual int activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3942).

```
6.484.3.6 virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3942).

```
6.484.3.7 virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3943).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h`

## 6.485 decaf::util::concurrent::Lock Class Reference

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

```
#include <src/main/decaf/util/concurrent/Lock.h>
```

#### Public Member Functions

- **Lock** (**Synchronizable** \*object, const bool initiallyLocked=true)  
*Constructor - initializes the object member and locks the object if desired.*
- virtual **~Lock** ()  
*Destructor - Unlocks the object if it is locked.*
- void **lock** ()  
*Locks the object.*
- void **unlock** ()  
*Unlocks the object if it is already locked, otherwise a call to this method has no effect.*
- bool **isLocked** () const

*Indicates whether or not the object is locked.*

### 6.485.1 Detailed Description

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

#### Since

1.0

### 6.485.2 Constructor & Destructor Documentation

**6.485.2.1** `decaf::util::concurrent::Lock ( Synchronizable * object, const bool initiallyLocked = true )`

Constructor - initializes the object member and locks the object if desired.

#### Parameters

<i>object</i>	The sync object to control
<i>initially-Locked</i>	If true, the object will automatically be locked.

**6.485.2.2** `virtual decaf::util::concurrent::Lock::~~Lock ( ) [virtual]`

Destructor - Unlocks the object if it is locked.

### 6.485.3 Member Function Documentation

**6.485.3.1** `bool decaf::util::concurrent::Lock::isLocked ( ) const [inline]`

Indicates whether or not the object is locked.

#### Returns

true if the object is locked, otherwise false.

**6.485.3.2** `void decaf::util::concurrent::Lock::lock ( )`

Locks the object.

**6.485.3.3** `void decaf::util::concurrent::Lock::unlock ( )`

Unlocks the object if it is already locked, otherwise a call to this method has no effect.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Lock.h`

## 6.486 decaf::util::concurrent::locks::Lock Class Reference

**Lock** (p. 2452) implementations provide more extensive locking operations than can be obtained using synchronized statements.

```
#include <src/main/decaf/util/concurrent/locks/Lock.h>
```

Inheritance diagram for `decaf::util::concurrent::locks::Lock`:

### Public Member Functions

- virtual `~Lock ()`
- virtual void **lock** ()=0 throw ( decaf::lang::exceptions::RuntimeException )  
*Acquires the lock.*
- virtual void **lockInterruptibly** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException )  
*Acquires the lock unless the current thread is interrupted.*
- virtual bool **tryLock** ()=0 throw ( decaf::lang::exceptions::RuntimeException )  
*Acquires the lock only if it is free at the time of invocation.*
- virtual bool **tryLock** (long long time, const **TimeUnit** &unit)=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException )  
*Acquires the lock if it is free within the given waiting time and the current thread has not been interrupted.*
- virtual void **unlock** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Releases the lock.*
- virtual **Condition** \* **newCondition** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::UnsupportedOperationException )  
*Returns a new **Condition** (p. 1282) instance that is bound to this **Lock** (p. 2452) instance.*

### 6.486.1 Detailed Description

**Lock** (p. 2452) implementations provide more extensive locking operations than can be obtained using synchronized statements. They allow more flexible structuring, may have quite different properties, and may support multiple associated **Condition** (p. 1282) objects.

A lock is a tool for controlling access to a shared resource by multiple threads. Commonly, a lock provides exclusive access to a shared resource: only one thread at a time can acquire the lock and all access to the shared resource requires that the lock be acquired first. However, some locks may allow concurrent access to a shared resource, such as the read lock of a **ReadWriteLock** (p. 3263).

While the scoping mechanism for synchronized statements makes it much easier to program with monitor locks, and helps avoid many common programming errors involving locks, there are occasions where you need to work with locks in a more flexible way. For example, some algorithms for traversing concurrently accessed data structures require the use of "hand-over-hand" or "chain locking": you acquire the lock of node A, then node B, then release A and acquire C, then release B and acquire D and so on. Implementations of the **Lock** (p. 2452) interface enable the use of such techniques by allowing a lock to be acquired and released in different scopes, and allowing multiple locks to be acquired and released in any order.

With this increased flexibility comes additional responsibility. The absence of block-structured locking removes the automatic release of locks that occurs with synchronized statements. In most cases, the following idiom should be used:

```
Lock (p. 2452) l = ...; l.lock(); try { // access the resource protected by this lock }  
catch(...) { l.unlock(); }
```

When locking and unlocking occur in different scopes, care must be taken to ensure that all code that is executed while the lock is held is protected by try-catch ensure that the lock is released when necessary.

**Lock** (p. 2452) implementations provide additional functionality over the use of synchronized methods and statements by providing a non-blocking attempt to acquire a lock (**tryLock()** (p. 2457)), an attempt to acquire the lock that can be interrupted (**lockInterruptibly()** (p. 2454)), and an attempt to acquire the lock that can timeout (**tryLock(long, TimeUnit)**).

Note that **Lock** (p. 2452) instances are just normal objects and can themselves be used as the target in a synchronized statement.

The three forms of lock acquisition (interruptible, non-interruptible, and timed) may differ in their performance characteristics, ordering guarantees, or other implementation qualities. Further, the ability to interrupt the ongoing acquisition of a lock may not be available in a given **Lock** (p. 2452) class. Consequently, an implementation is not required to define exactly the same guarantees or semantics for all three forms of lock acquisition, nor is it required to support interruption of an ongoing lock acquisition. An implementation is required to clearly document the semantics and guarantees provided by each of the locking methods. It must also obey the interruption semantics as defined in this interface, to the extent that interruption of lock acquisition is supported: which is either totally, or only on method entry.

As interruption generally implies cancellation, and checks for interruption are often infrequent, an implementation can favor responding to an interrupt over normal method return. This is true even if it can be shown that the interrupt occurred after another action may have unblocked the thread. An implementation should document this behavior.

#### Since

1.0

### 6.486.2 Constructor & Destructor Documentation

6.486.2.1 `virtual decaf::util::concurrent::locks::Lock::~~Lock( ) [inline, virtual]`

### 6.486.3 Member Function Documentation

6.486.3.1 `virtual void decaf::util::concurrent::locks::Lock::lock( ) throw ( decaf::lang::exceptions::RuntimeException ) [pure virtual]`

Acquires the lock.

If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until the lock has been acquired.

#### Implementation Considerations

A **Lock** (p. 2452) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p. 2452) implementation.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
-------------------------	--

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 3276).

6.486.3.2 `virtual void decaf::util::concurrent::locks::Lock::lockInterruptibly( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException ) [pure virtual]`

Acquires the lock unless the current thread is interrupted.

Acquires the lock if it is available and returns immediately.

If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

\* The lock is acquired by the current thread; or \* Some other thread interrupts the current thread, and interruption of lock acquisition is supported.

If the current thread:



\* has its interrupted status set on entry to this method; or \* is interrupted while acquiring the lock, and interruption of lock acquisition is supported,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

#### Implementation Considerations

The ability to interrupt a lock acquisition in some implementations may not be possible, and if possible may be an expensive operation. The programmer should be aware that this may be the case. An implementation should document when this is the case.

An implementation can favor responding to an interrupt over normal method return.

A **Lock** (p. 2452) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p. 2452) implementation.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
<i>InterruptedException</i>	if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported).

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 3276).

**6.486.3.3** `virtual Condition* decaf::util::concurrent::locks::Lock::newCondition  
( ) throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::UnsupportedOperationException ) [pure  
virtual]`

Returns a new **Condition** (p. 1282) instance that is bound to this **Lock** (p. 2452) instance.

Before waiting on the condition the lock must be held by the current thread. A call to **Condition.await()** (p. 1285) will atomically release the lock before waiting and re-acquire the lock before the wait returns.

#### Implementation Considerations

The exact operation of the **Condition** (p. 1282) instance depends on the **Lock** (p. 2452) implementation and must be documented by that implementation.

#### Returns

A new **Condition** (p. 1282) instance for this **Lock** (p. 2452) instance the caller must delete the returned **Condition** (p. 1282) object when done with it.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while creating the <b>Condition</b> (p. 1282).
<i>UnsupportedOperationException</i>	if this <b>Lock</b> (p. 2452) implementation does not support conditions

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 3277).

**6.486.3.4** `virtual bool decaf::util::concurrent::locks::Lock::tryLock ( long long time, const TimeUnit & unit ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException ) [pure virtual]`

Acquires the lock if it is free within the given waiting time and the current thread has not been interrupted.

If the lock is available this method returns immediately with the value true. If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

\* The lock is acquired by the current thread; or \* Some other thread interrupts the current thread, and interruption of lock acquisition is supported; or \* The specified waiting time elapses

If the lock is acquired then the value true is returned.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while acquiring the lock, and interruption of lock acquisition is supported,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

#### Implementation Considerations

The ability to interrupt a lock acquisition in some implementations may not be possible, and if possible may be an expensive operation. The programmer should be aware that this may be the case. An implementation should document when this is the case.

An implementation can favor responding to an interrupt over normal method return, or reporting a timeout.

A **Lock** (p. 2452) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an (unchecked) exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p. 2452) implementation.

#### Parameters

<i>time</i>	the maximum time to wait for the lock
<i>unit</i>	the time unit of the time argument

#### Returns

true if the lock was acquired and false if the waiting time elapsed before the lock was acquired

#### Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
-------------------------	--

<i>InterruptedException</i>	if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported)
-----------------------------	---

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 3278).

**6.486.3.5** `virtual bool decaf::util::concurrent::locks::Lock::tryLock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [pure virtual]`

Acquires the lock only if it is free at the time of invocation.

Acquires the lock if it is available and returns immediately with the value true. If the lock is not available then this method will return immediately with the value false.

A typical usage idiom for this method would be:

**Lock** (p. 2452) `lock = ...; if (lock.tryLock()) { try { // manipulate protected state } catch(...) { lock.unlock(); } } else { // perform alternative actions }`

This usage ensures that the lock is unlocked if it was acquired, and doesn't try to unlock if the lock was not acquired.

### Returns

true if the lock was acquired and false otherwise

### Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
-------------------------	--

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 3279).

**6.486.3.6** `virtual void decaf::util::concurrent::locks::Lock::unlock ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException ) [pure virtual]`

Releases the lock.

### Implementation Considerations

A **Lock** (p. 2452) implementation will usually impose restrictions on which thread can release a lock (typically only the holder of the lock can release it) and may throw an exception if the restriction is violated. Any restrictions and the exception type must be documented by that **Lock** (p. 2452) implementation.

### Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
<i>IllegalMonitorStateException</i>	if the current thread is not the owner of the lock.

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 3280).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/Lock.h`

## 6.487 decaf::util::concurrent::locks::LockSupport Class Reference

Basic thread blocking primitives for creating locks and other synchronization classes.

```
#include <src/main/decaf/util/concurrent/locks/LockSupport.h>
```

### Public Member Functions

- `~LockSupport ()`

### Static Public Member Functions

- static void **unpark** (**decaf::lang::Thread** \*thread) throw ()  
*Makes available the permit for the given thread, if it was not already available.*
- static void **park** () throw ()  
*Disables the current thread for thread scheduling purposes unless the permit is available.*
- static void **parkNanos** (long long nanos) throw ()  
*Disables the current thread for thread scheduling purposes, for up to the specified waiting time, unless the permit is available.*
- static void **parkUntil** (long long deadline) throw ()  
*Disables the current thread for thread scheduling purposes, until the specified deadline, unless the permit is available.*

### 6.487.1 Detailed Description

Basic thread blocking primitives for creating locks and other synchronization classes. This class associates, with each thread that uses it, a permit (in the sense of the **Semaphore** (p. 3434) class). A call to `park` will return immediately if the permit is available, consuming it in the process; otherwise it may block. A call to `unpark` makes the permit available, if it was not already available. (Unlike with Semaphores though, permits do not accumulate. There is at most one.)

Methods `park` and `unpark` provide efficient means of blocking and unblocking threads. Races between one thread invoking `park` and another thread trying to `unpark` it will preserve liveness, due to the permit. Additionally, `park` will return if the caller's thread was interrupted, and timeout versions are supported. The `park` method may also return

at any other time, for "no reason", so in general must be invoked within a loop that rechecks conditions upon return. In this sense park serves as an optimization of a "busy wait" that does not waste as much time spinning, but must be paired with an unpark to be effective.

These methods are designed to be used as tools for creating higher-level synchronization utilities, and are not in themselves useful for most concurrency control applications. The park method is designed for use only in constructions of the form:

```
while (!canProceed()) { ... LockSupport.park(this); }
```

where neither canProceed nor any other actions prior to the call to park entail locking or blocking. Because only one permit is associated with each thread, any intermediary uses of park could interfere with its intended effects.

Sample Usage. Here is a sketch of a first-in-first-out non-reentrant lock class:

```
class FIFOMutex { private:
    AtomicBoolean locked; ConcurrentLinkedQueue<Thread*> waiters;
public:
    void lock() {
        bool wasInterrupted = false; Thread* current = Thread::currentThread(); waiters.add(
            current );
        // Block while not first in queue or cannot acquire lock while( waiters.peek() != current
        || !locked.compareAndSet( false, true ) ) {
            LockSupport.park(this); if( Thread::interrupted() ) // ignore interrupts while waiting
            wasInterrupted = true; }
        waiters.remove(); if( wasInterrupted ) // reassert interrupt status on exit current.interrupt();
    }
    void unlock() { locked.set( false ); LockSupport.unpark (p. 2461)( waiters.peek() );
    } };
```

Since

1.0

## 6.487.2 Constructor & Destructor Documentation

6.487.2.1 decaf::util::concurrent::locks::LockSupport::~~LockSupport ( )

## 6.487.3 Member Function Documentation

6.487.3.1 static void decaf::util::concurrent::locks::LockSupport::park ( ) throw ()  
[static]

Disables the current thread for thread scheduling purposes unless the permit is available.

If the permit is available then it is consumed and the call returns immediately; otherwise the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

- \* Some other thread invokes `unpark` with the current thread as the target; or
- \* Some other thread interrupts the current thread; or
- \* The call spuriously (that is, for no reason) returns.

This method does not report which of these caused the method to return. Callers should re-check the conditions which caused the thread to park in the first place. Callers may also determine, for example, the interrupt status of the thread upon return.

**6.487.3.2** `static void decaf::util::concurrent::locks::LockSupport::parkNanos ( long long nanos ) throw () [static]`

Disables the current thread for thread scheduling purposes, for up to the specified waiting time, unless the permit is available.

If the permit is available then it is consumed and the call returns immediately; otherwise the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

- \* Some other thread invokes `unpark` with the current thread as the target; or
- \* Some other thread interrupts the current thread; or
- \* The specified waiting time elapses; or
- \* The call spuriously (that is, for no reason) returns.

This method does not report which of these caused the method to return. Callers should re-check the conditions which caused the thread to park in the first place. Callers may also determine, for example, the interrupt status of the thread, or the elapsed time upon return.

#### Parameters

<i>nanos</i>	the maximum number of nanoseconds to wait
--------------	---

**6.487.3.3** `static void decaf::util::concurrent::locks::LockSupport::parkUntil ( long long deadline ) throw () [static]`

Disables the current thread for thread scheduling purposes, until the specified deadline, unless the permit is available.

If the permit is available then it is consumed and the call returns immediately; otherwise the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

- \* Some other thread invokes `unpark` with the current thread as the target; or
- \* Some other thread interrupts the current thread; or
- \* The specified deadline passes; or
- \* The call spuriously (that is, for no reason) returns.

This method does not report which of these caused the method to return. Callers should re-check the conditions which caused the thread to park in the first place. Callers may also determine, for example, the interrupt status of the thread, or the current time upon

return.

#### Parameters

<i>deadline</i>	the absolute time, in milliseconds from the Epoch, to wait until
-----------------	--

**6.487.3.4** static void decaf::util::concurrent::locks::LockSupport::unpark ( decaf::lang::Thread \* *thread* ) throw () [static]

Makes available the permit for the given thread, if it was not already available.

If the thread was blocked on park then it will unblock. Otherwise, its next call to park is guaranteed not to block. This operation is not guaranteed to have any effect at all if the given thread has not been started.

#### Parameters

<i>thread</i>	the thread to unport, or NULL in which case the method has no effect.
---------------	---

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/locks/**LockSupport.h**

## 6.488 decaf::util::logging::Logger Class Reference

A **Logger** (p. 2461) object is used to log messages for a specific system or application component.

```
#include <src/main/decaf/util/logging/Logger.h>
```

### Public Member Functions

- virtual ~**Logger** ()
- const std::string & **getName** () const  
*Gets the name of this **Logger** (p. 2461).*
- **Logger** \* **getParent** () const  
*Gets the parent of this **Logger** (p. 2461) which will be the nearest existing **Logger** (p. 2461) in this Loggers namespace.*
- void **setParent** (**Logger** \*parent)  
*Set (p. 3538) the parent for this **Logger** (p. 2461).*
- void **addHandler** (**Handler** \*handler) throw ( lang::exceptions::NullPointerException )  
*Add a log **Handler** (p. 2042) to receive logging messages.*

- void **removeHandler** (**Handler** \*handler)  
*Removes the specified **Handler** (p. 2042) from this logger, ownership of the **Handler** (p. 2042) pointer is returned to the caller.*
- const std::list< **Handler** \* > & **getHandlers** () const  
*Gets a vector containing all the handlers that this class has been assigned to use.*
- void **setFilter** (**Filter** \*filter)  
*Set (p. 3538) a filter to control output on this **Logger** (p. 2461).*
- const **Filter** \* **getFilter** () const  
*Gets the **Filter** (p. 1949) object that this class is using.*
- **Level** **getLevel** () const  
*Get the log **Level** (p. 2403) that has been specified for this **Logger** (p. 2461).*
- void **setLevel** (const **Level** &level)  
*Set (p. 3538) the log level specifying which message levels will be logged by this logger.*
- bool **getUseParentHandlers** () const  
*Discover whether or not this logger is sending its output to its parent logger.*
- void **setUseParentHandlers** (bool value)  
*Specify whether or not this logger should send its output to it's parent **Logger** (p. 2461).*
- virtual void **entering** (const std::string &blockName, const std::string &file, const int line)  
*Logs an Block Enter message.*
- virtual void **exiting** (const std::string &blockName, const std::string &file, const int line)  
*Logs an Block Exit message.*
- virtual void **severe** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a SEVERE **Level** (p. 2403) Log.*
- virtual void **warning** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a WARN **Level** (p. 2403) Log.*
- virtual void **info** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a INFO **Level** (p. 2403) Log.*



- virtual void **debug** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a **DEBUG Level** (p. 2403) Log.*
- virtual void **config** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a **CONFIG Level** (p. 2403) Log.*
- virtual void **fine** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a **FINE Level** (p. 2403) Log.*
- virtual void **finer** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a **FINER Level** (p. 2403) Log.*
- virtual void **finest** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a **FINEST Level** (p. 2403) Log.*
- virtual void **throwing** (const std::string &file, const int line, const std::string functionName, const decaf::lang::Throwable &thrown)  
*Log throwing an exception.*
- virtual bool **isLoggable** (const Level &level) const  
*Check if a message of the given level would actually be logged by this logger.*
- virtual void **log** (LogRecord &record)  
*Log a **LogRecord** (p. 2487).*
- virtual void **log** (const Level &level, const std::string &message)  
*Log a message, with no arguments.*
- virtual void **log** (const Level &levels, const std::string &file, const int line, const std::string &message,...)  
*Log a message, with the list of params that is formatted into the message string.*
- virtual void **log** (const Level &level, const std::string &file, const int line, const std::string &message, lang::Exception &ex)  
*Log a message, with associated Throwable information.*

### Static Public Member Functions

- static Logger \* **getAnonymousLogger** ()  
*Creates an anonymous logger.*

- static **Logger** \* **getLogger** (const std::string &name)

*Find or create a logger for a named subsystem.*

## Protected Member Functions

- **Logger** (const std::string &name)

*Creates a new instance of the **Logger** (p. 2461) with the given name.*

### 6.488.1 Detailed Description

A **Logger** (p. 2461) object is used to log messages for a specific system or application component. Loggers are normally named, using a hierarchical dot-separated namespace. **Logger** (p. 2461) names can be arbitrary strings, but they should normally be based on the namespace or class name of the logged component, such as **decaf.net** (p. 145) or **org.apache.decaf**. In addition it is possible to create "anonymous" Loggers that are not stored in the **Logger** (p. 2461) namespace.

**Logger** (p. 2461) objects may be obtained by calls on one of the **getLogger** factory methods. These will either create a new **Logger** (p. 2461) or return a suitable existing **Logger** (p. 2461).

Logging messages will be forwarded to registered **Handler** (p. 2042) objects, which can forward the messages to a variety of destinations, including consoles, files, OS logs, etc.

Each **Logger** (p. 2461) keeps track of a "parent" **Logger** (p. 2461), which is its nearest existing ancestor in the **Logger** (p. 2461) namespace.

Each **Logger** (p. 2461) has a "Level" associated with it. This reflects a minimum **Level** (p. 2403) that this logger cares about. If a Logger's level is set to **Level::INHERIT** (p. 2408), then its effective level is inherited from its parent, which may in turn obtain it recursively from its parent, and so on up the tree.

The log level can be configured based on the properties from the logging configuration file, as described in the description of the **LogManager** (p. 2480) class. However it may also be dynamically changed by calls on the **Logger.setLevel** (p. 2472) method. If a logger's level is changed the change may also affect child loggers, since any child logger that has 'inherit' as its level will inherit its effective level from its parent.

On each logging call the **Logger** (p. 2461) initially performs a cheap check of the request level (e.g. SEVERE or FINE) against the effective log level of the logger. If the request level is lower than the log level, the logging call returns immediately.

After passing this initial (cheap) test, the **Logger** (p. 2461) will allocate a **LogRecord** (p. 2487) to describe the logging message. It will then call a **Filter** (p. 1949) (if present) to do a more detailed check on whether the record should be published. If that passes it will then publish the **LogRecord** (p. 2487) to its output Handlers. By default, loggers also publish to their parent's Handlers, recursively up the tree.

Formatting is the responsibility of the output **Handler** (p. 2042), which will typically call a **Formatter** (p. 2027).

Note that formatting need not occur synchronously. It may be delayed until a **LogRecord** (p. 2487) is actually written to an external sink.

All methods on **Logger** (p. 2461) are thread safe.

#### Since

1.0

### 6.488.2 Constructor & Destructor Documentation

#### 6.488.2.1 decaf::util::logging::Logger::Logger ( const std::string & name ) [protected]

Creates a new instance of the **Logger** (p. 2461) with the given name.

The logger will be initially configured with a null **Level** (p. 2403) and with useParentHandlers true.

#### Parameters

<i>name</i>	A name for the logger. This should be a dot-separated name and should normally be based on the package name or class name of the subsystem, such as <b>decaf.net</b> (p. 145) or org.apache.decaf. It may be empty for anonymous Loggers.
-------------	---

#### 6.488.2.2 virtual decaf::util::logging::Logger::~~Logger ( ) [virtual]

### 6.488.3 Member Function Documentation

#### 6.488.3.1 void decaf::util::logging::Logger::addHandler ( Handler \* handler ) throw ( lang::exceptions::NullPointerException )

Add a log **Handler** (p. 2042) to receive logging messages.

By default, Loggers also send their output to their parent logger. Typically the root **Logger** (p. 2461) is configured with a set of Handlers that essentially act as default handlers for all loggers.

The ownership of the given **Handler** (p. 2042) is passed to the **Logger** (p. 2461) and the **Handler** (p. 2042) will be deleted when this **Logger** (p. 2461) is destroyed unless the caller first calls removeHandler with the same pointer value as was originally given.

#### Parameters

<i>handler</i>	A Logging <b>Handler</b> (p. 2042)
----------------	------------------------------------

**Exceptions**

<i>NullPointerException</i>	if the <b>Handler</b> (p. 2042) given is NULL.
-----------------------------	--

**6.488.3.2** `virtual void decaf::util::logging::Logger::config ( const std::string & file, const int line,  
const std::string functionName, const std::string & message )` [virtual]

Log a CONFIG **Level** (p. 2403) Log.

If the logger is currently enabled for the CONFIG message level then the given message is forwarded to all the registered output **Handler** (p. 2042) objects.

**Parameters**

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this <b>Level</b> (p. 2403).

**6.488.3.3** `virtual void decaf::util::logging::Logger::debug ( const std::string & file, const int line,  
const std::string functionName, const std::string & message )` [virtual]

Log a DEBUG **Level** (p. 2403) Log.

If the logger is currently enabled for the DEBUG message level then the given message is forwarded to all the registered output **Handler** (p. 2042) objects.

**Parameters**

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this <b>Level</b> (p. 2403).

**6.488.3.4** `virtual void decaf::util::logging::Logger::entering ( const std::string & blockName,  
const std::string & file, const int line )` [virtual]

Logs an Block Enter message.

This is a convenience method that is used to tag a block enter, a log record with the given information is logged at the **Level::FINER** (p. 2408) log level.

**Parameters**

<i>blockName</i>	The source block name, (usually ClassName::MethodName, or Method-Name).
------------------	---

<i>file</i>	The source file name where this method was called from.
<i>line</i>	The source line number where this method was called from.

**6.488.3.5** `virtual void decaf::util::logging::Logger::exiting ( const std::string & blockName,  
const std::string & file, const int line )` [virtual]

Logs an Block Exit message.

This is a convenience method that is used to tag a block enter, a log record with the given information is logged at the **Level::FINER** (p. 2408) log level.

#### Parameters

<i>blockName</i>	The source block name, (usually <code>ClassName::MethodName</code> , or <code>Method-Name</code> ).
<i>file</i>	The source file name where this method was called from.
<i>line</i>	The source line number where this method was called from.

**6.488.3.6** `virtual void decaf::util::logging::Logger::fine ( const std::string & file, const int line,  
const std::string functionName, const std::string & message )` [virtual]

Log a FINE **Level** (p. 2403) Log.

If the logger is currently enabled for the FINE message level then the given message is forwarded to all the registered output **Handler** (p. 2042) objects.

#### Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this <b>Level</b> (p. 2403).

**6.488.3.7** `virtual void decaf::util::logging::Logger::finer ( const std::string & file, const int line,  
const std::string functionName, const std::string & message )` [virtual]

Log a FINER **Level** (p. 2403) Log.

If the logger is currently enabled for the FINER message level then the given message is forwarded to all the registered output **Handler** (p. 2042) objects.

#### Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.

<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this <b>Level</b> (p. 2403).

**6.488.3.8** `virtual void decaf::util::logging::Logger::finest ( const std::string & file, const int line, const std::string functionName, const std::string & message ) [virtual]`

Log a **FINEST Level** (p. 2403) Log.

If the logger is currently enabled for the **FINEST** message level then the given message is forwarded to all the registered output **Handler** (p. 2042) objects.

#### Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this <b>Level</b> (p. 2403).

**6.488.3.9** `static Logger* decaf::util::logging::Logger::getAnonymousLogger ( ) [static]`

Creates an anonymous logger.

The newly created **Logger** (p. 2461) is not registered in the **LogManager** (p. 2480) namespace. There will be no access checks on updates to the logger. Even although the new logger is anonymous, it is configured to have the root logger ("" ) as its parent. This means that by default it inherits its effective level and handlers from the root logger.

The caller is responsible for destroying the returned logger.

#### Returns

Newly created anonymous logger

**6.488.3.10** `const Filter* decaf::util::logging::Logger::getFilter ( ) const [inline]`

Gets the **Filter** (p. 1949) object that this class is using.

#### Returns

the **Filter** (p. 1949) in use, (can be NULL).

**6.488.3.11** `const std::list<Handler*>& decaf::util::logging::Logger::getHandlers ( ) const`

Gets a vector containing all the handlers that this class has been assigned to use.

**Returns**

a list of handlers that are used by this logger

**6.488.3.12** `Level decaf::util::logging::Logger::getLevel ( ) const` `[inline]`

Get the log **Level** (p. 2403) that has been specified for this **Logger** (p. 2461).

The result may be the INHERIT level, which means that this logger's effective level will be inherited from its parent.

**Returns**

the level that is currently set

**6.488.3.13** `static Logger* decaf::util::logging::Logger::getLogger ( const std::string & name )`  
`[static]`

Find or create a logger for a named subsystem.

If a logger has already been created with the given name it is returned. Otherwise a new logger is created.

If a new logger is created its log level will be configured based on the **LogManager** (p. 2480) and it will be configured to also send logging output to its parent loggers Handlers. It will be registered in the **LogManager** (p. 2480) global namespace.

**Parameters**

<i>name</i>	- A name for the logger. This should be a dot-separated name and should normally be based on the package name or class name of the subsystem, such as cms or <b>activemq.core.ActiveMQConnection</b> (p. 260)
-------------	---

**Returns**

a suitable logger.

**6.488.3.14** `const std::string& decaf::util::logging::Logger::getName ( ) const` `[inline]`

Gets the name of this **Logger** (p. 2461).

**Returns**

logger name

**6.488.3.15** `Logger* decaf::util::logging::Logger::getParent ( ) const` `[inline]`

Gets the parent of this **Logger** (p. 2461) which will be the nearest existing **Logger** (p. 2461) in this Loggers namespace.

If this is the Root **Logger** (p. 2461) than this method returns NULL.

#### Returns

Pointer to this Loggers nearest parent **Logger** (p. 2461).

**6.488.3.16** `bool decaf::util::logging::Logger::getUseParentHandlers ( ) const` `[inline]`

Discover whether or not this logger is sending its output to its parent logger.

#### Returns

true if using Parent Handlers

**6.488.3.17** `virtual void decaf::util::logging::Logger::info ( const std::string & file, const int line, const std::string functionName, const std::string & message )` `[virtual]`

Log a **INFO Level** (p. 2403) Log.

If the logger is currently enabled for the INFO message level then the given message is forwarded to all the registered output **Handler** (p. 2042) objects.

#### Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this <b>Level</b> (p. 2403).

**6.488.3.18** `virtual bool decaf::util::logging::Logger::isLoggable ( const Level & level ) const` `[virtual]`

Check if a message of the given level would actually be logged by this logger.

This check is based on the Loggers effective level, which may be inherited from its parent.

#### Parameters

<i>level</i>	- a message logging level
--------------	---------------------------



**Returns**

true if the given message level is currently being logged.

**6.488.3.19** virtual void decaf::util::logging::Logger::log ( const Level & *levels*, const std::string & *file*, const int *line*, const std::string & *message*, ... ) [virtual]

Log a message, with the list of params that is formatted into the message string.

If the logger is currently enabled for the given message level then the given message is forwarded to all the registered output **Handler** (p. 2042) objects

**Parameters**

<i>level</i>	the <b>Level</b> (p. 2403) to log at
<i>file</i>	the message to log
<i>line</i>	the line in the file
...	variable length argument to format the message string.

**6.488.3.20** virtual void decaf::util::logging::Logger::log ( LogRecord & *record* ) [virtual]

Log a **LogRecord** (p. 2487).

All the other logging methods in this class call through this method to actually perform any logging. Subclasses can override this single method to capture all log activity.

**Parameters**

<i>record</i>	- the <b>LogRecord</b> (p. 2487) to be published
---------------	--

**6.488.3.21** virtual void decaf::util::logging::Logger::log ( const Level & *level*, const std::string & *message* ) [virtual]

Log a message, with no arguments.

If the logger is currently enabled for the given message level then the given message is forwarded to all the registered output **Handler** (p. 2042) objects

**Parameters**

<i>level</i>	the <b>Level</b> (p. 2403) to log at
<i>message</i>	the message to log

**6.488.3.22** `virtual void decaf::util::logging::Logger::log ( const Level & level, const std::string & file, const int line, const std::string & message, lang::Exception & ex )`  
`[virtual]`

Log a message, with associated Throwable information.

If the logger is currently enabled for the given message level then the given arguments are stored in a **LogRecord** (p. 2487) which is forwarded to all registered output handlers. Note that the thrown argument is stored in the **LogRecord** (p. 2487) thrown property, rather than the **LogRecord** (p. 2487) parameters property. Thus is it processed specially by output Formatters and is not treated as a formatting parameter to the **LogRecord** (p. 2487) message property.

#### Parameters

<i>level</i>	the <b>Level</b> (p. 2403) to log at.
<i>file</i>	File that the message was logged in.
<i>line</i>	the line number where the message was logged at.
<i>message</i>	the message to log.
<i>ex</i>	the Exception to log

**6.488.3.23** `void decaf::util::logging::Logger::removeHandler ( Handler * handler )`

Removes the specified **Handler** (p. 2042) from this logger, ownership of the **Handler** (p. 2042) pointer is returned to the caller.

Returns silently if the given **Handler** (p. 2042) is not found.

#### Parameters

<i>handler</i>	The <b>Handler</b> (p. 2042) to remove
----------------	--

**6.488.3.24** `void decaf::util::logging::Logger::setFilter ( Filter * filter )`

**Set** (p. 3538) a filter to control output on this **Logger** (p. 2461).

After passing the initial "level" check, the **Logger** (p. 2461) will call this **Filter** (p. 1949) to check if a log record should really be published.

The caller releases ownership of this filter to this logger

#### Parameters

<i>filter</i>	The <b>Filter</b> (p. 1949) to use, (can be NULL).
---------------	--

**6.488.3.25** `void decaf::util::logging::Logger::setLevel ( const Level & level )` `[inline]`

**Set** (p. 3538) the log level specifying which message levels will be logged by this logger.

Message levels lower than this value will be discarded. The level value **Level::OFF** (p. 2408) can be used to turn off logging.

If the new level is the **INHERIT Level** (p. 2403), it means that this node should inherit its level from its nearest ancestor with a specific (non-**INHERIT**) level value.

#### Parameters

<i>level</i>	The new <b>Level</b> (p. 2403) value to use when logging.
--------------	---

**6.488.3.26** void decaf::util::logging::Logger::setParent ( **Logger** \* *parent* ) [inline]

**Set** (p. 3538) the parent for this **Logger** (p. 2461).

This method is used by the **LogManager** (p. 2480) to update a **Logger** (p. 2461) when the namespace changes.

It should not be called from application code.

**6.488.3.27** void decaf::util::logging::Logger::setUseParentHandlers ( bool *value* ) [inline]

Specify whether or not this logger should send its output to it's parent **Logger** (p. 2461).

This means that any LogRecords will also be written to the parent's Handlers, and potentially to its parent, recursively up the namespace.

#### Parameters

<i>value</i>	True is output is to be written to the parent
--------------	---

**6.488.3.28** virtual void decaf::util::logging::Logger::severe ( const std::string & *file*, const int *line*, const std::string *functionName*, const std::string & *message* ) [virtual]

Log a **SEVERE Level** (p. 2403) Log.

If the logger is currently enabled for the **SEVERE** message level then the given message is forwarded to all the registered output **Handler** (p. 2042) objects.

#### Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this <b>Level</b> (p. 2403).

**6.488.3.29** `virtual void decaf::util::logging::Logger::throwing ( const std::string & file, const int line, const std::string functionName, const decaf::lang::Throwable & thrown )`  
[virtual]

Log throwing an exception.

This is a convenience method to log that a method is terminating by throwing an exception. The logging is done using the FINER level.

If the logger is currently enabled for the given message level then the given arguments are stored in a **LogRecord** (p. 2487) which is forwarded to all registered output handlers. The LogRecord's message is set to "THROW".

#### Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>thrown</i>	The Throwable that will be thrown, will be cloned.

**6.488.3.30** `virtual void decaf::util::logging::Logger::warning ( const std::string & file, const int line, const std::string functionName, const std::string & message )` [virtual]

Log a WARN **Level** (p. 2403) Log.

If the logger is currently enabled for the WARN message level then the given message is forwarded to all the registered output **Handler** (p. 2042) objects.

#### Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this <b>Level</b> (p. 2403).

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Logger.h**

## 6.489 decaf::util::logging::LoggerHierarchy Class Reference

```
#include <src/main/decaf/util/logging/LoggerHierarchy.h>
```

### Public Member Functions

- **LoggerHierarchy** ()

- virtual `~LoggerHierarchy()`

### 6.489.1 Constructor & Destructor Documentation

6.489.1.1 `decaf::util::logging::LoggerHierarchy::LoggerHierarchy()`

6.489.1.2 `virtual decaf::util::logging::LoggerHierarchy::~LoggerHierarchy()` `[virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/LoggerHierarchy.h`

## 6.490 activemq::io::LoggingInputStream Class Reference

```
#include <src/main/activemq/io/LoggingInputStream.h>
```

Inheritance diagram for `activemq::io::LoggingInputStream`:

### Public Member Functions

- `LoggingInputStream(decaf::io::InputStream *inputStream, bool own=false)`

*Creates a DataInputStream that uses the specified underlying InputStream.*

- virtual `~LoggingInputStream()`

### Protected Member Functions

- virtual `int doReadByte()` `throw (decaf::io::IOException)`
- virtual `int doReadArrayBounded(unsigned char *buffer, int size, int offset, int length) throw (decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException)`

### 6.490.1 Constructor & Destructor Documentation

6.490.1.1 `activemq::io::LoggingInputStream::LoggingInputStream(decaf::io::InputStream *inputStream, bool own=false)`

Creates a DataInputStream that uses the specified underlying InputStream.

#### Parameters

<i>inputStream</i>	the InputStream instance to wrap.
<i>own</i>	indicates if this class owns the wrapped string defaults to false.

6.490.1.2 virtual `activemq::io::LoggingInputStream::~~LoggingInputStream ( )`  
[virtual]

## 6.490.2 Member Function Documentation

6.490.2.1 virtual int `activemq::io::LoggingInputStream::doReadArrayBounded ( unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )` [protected, virtual]

Reimplemented from `decaf::io::FilterInputStream` (p. 1954).

6.490.2.2 virtual int `activemq::io::LoggingInputStream::doReadByte ( ) throw ( decaf::io::IOException )` [protected, virtual]

Reimplemented from `decaf::io::FilterInputStream` (p. 1954).

The documentation for this class was generated from the following file:

- `src/main/activemq/io/LoggingInputStream.h`

## 6.491 activemq::io::LoggingOutputStream Class Reference

OutputStream filter that just logs the data being written.

```
#include <src/main/activemq/io/LoggingOutputStream.h>
```

Inheritance diagram for `activemq::io::LoggingOutputStream`:

### Public Member Functions

- **LoggingOutputStream** (OutputStream \*next, bool own=false)  
*Constructor.*
- virtual `~LoggingOutputStream ( )`

### Protected Member Functions

- virtual void **doWriteByte** (unsigned char c) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### 6.491.1 Detailed Description

OutputStream filter that just logs the data being written.

### 6.491.2 Constructor & Destructor Documentation

**6.491.2.1** `activemq::io::LoggingOutputStream::LoggingOutputStream ( OutputStream * next, bool own = false )`

Constructor.

#### Parameters

<i>next</i>	The OutputStream to wrap an write logs to.
<i>own</i>	If true, this object will control the lifetime of the output stream that it encapsulates.

**6.491.2.2** `virtual activemq::io::LoggingOutputStream::~~LoggingOutputStream ( )`  
[virtual]

### 6.491.3 Member Function Documentation

**6.491.3.1** `virtual void activemq::io::LoggingOutputStream::doWriteArrayBounded ( const unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`  
[protected, virtual]

Reimplemented from `decaf::io::FilterOutputStream` (p. 1960).

**6.491.3.2** `virtual void activemq::io::LoggingOutputStream::doWriteByte ( unsigned char c )`  
`throw ( decaf::io::IOException )` [protected, virtual]

Reimplemented from `decaf::io::FilterOutputStream` (p. 1960).

The documentation for this class was generated from the following file:

- `src/main/activemq/io/LoggingOutputStream.h`

## 6.492 activemq::transport::logging::LoggingTransport Class Reference

A transport filter that logs commands as they are sent/received.

```
#include <src/main/activemq/transport/logging/LoggingTransport.h>
```

Inheritance diagram for `activemq::transport::logging::LoggingTransport`:

## Public Member Functions

- **LoggingTransport** (const **Pointer**< **Transport** > &next)

*Constructor.*

- virtual **~LoggingTransport** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)

*Event handler for the receipt of a command.*

- virtual void **oneway** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Sends a one-way command.*

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Not supported by this class - throws an exception.*

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Not supported by this class - throws an exception.*

### 6.492.1 Detailed Description

A transport filter that logs commands as they are sent/received.

### 6.492.2 Constructor & Destructor Documentation

#### 6.492.2.1 `activemq::transport::logging::LoggingTransport::LoggingTransport ( const Pointer< Transport > &next )`

Constructor.

#### Parameters

<i>next</i>	- the next <b>Transport</b> (p. 3996) in the chain
-------------	--



6.492.2.2 `virtual activemq::transport::logging::LoggingTransport::~LoggingTransport ( )`  
`[inline, virtual]`

### 6.492.3 Member Function Documentation

6.492.3.1 `virtual void activemq::transport::logging::LoggingTransport::onCommand ( const  
 Pointer< Command > & command )` `[virtual]`

Event handler for the receipt of a command.

#### Parameters

<i>command</i>	- the received command object.
----------------	--------------------------------

Reimplemented from `activemq::transport::TransportFilter` (p. 4009).

6.492.3.2 `virtual void activemq::transport::logging::LoggingTransport::oneway ( const  
 Pointer< Command > & command ) throw ( decaf::io::IOException,  
 decaf::lang::exceptions::UnsupportedOperationException )`  
`[virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

#### Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

#### Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Reimplemented from `activemq::transport::TransportFilter` (p. 4010).

6.492.3.3 `virtual Pointer<Response> activemq::transport::logging::LoggingTransport::request ( const  
 Pointer< Command > & command ) throw ( decaf::io::IOException,  
 decaf::lang::exceptions::UnsupportedOperationException )`  
`[virtual]`

Not supported by this class - throws an exception.

#### Parameters

<i>command</i>	the command that is sent as a request
----------------	---------------------------------------

**Exceptions**

<i>UnsupportedOperation</i> <i>Exception.</i>	
--	--

Reimplemented from **activemq::transport::TransportFilter** (p. 4011).

**6.492.3.4** virtual **Pointer<Response>** **activemq::transport::logging::LoggingTransport::request** (  
**const Pointer< Command > & command**, unsigned  
**int timeout** ) throw ( **decaf::io::IOException**, **de-**  
**caf::lang::exceptions::UnsupportedOperationException** )  
 [virtual]

Not supported by this class - throws an exception.

**Parameters**

<i>command</i>	the command that is sent as a request
<i>timeout</i>	the time to wait for a response.

**Exceptions**

<i>UnsupportedOperation</i> <i>Exception.</i>	
--	--

Reimplemented from **activemq::transport::TransportFilter** (p. 4011).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/logging/LoggingTransport.h`

**6.493 decaf::util::logging::LogManager Class Reference**

There is a single global **LogManager** (p. 2480) object that is used to maintain a set of shared state about Loggers and log services.

```
#include <src/main/decaf/util/logging/LogManager.h>
```

**Public Member Functions**

- virtual **~LogManager** ()
- bool **addLogger** (**Logger** \*logger) throw ( **decaf::lang::exceptions::NullPointerException**, **decaf::lang::exceptions::IllegalArgumentException** )  
*Add a named logger.*
- **Logger** \* **getLogger** (const std::string &name)

Retrieves or creates a new **Logger** (p. 2461) using the name specified a new logger inherits the configuration of the logger's parent if there is no configuration data for the logger.

- **int getLoggerNames** (const std::vector< std::string > &names)

*Gets a list of known **Logger** (p. 2461) Names from this Manager, new loggers added while this method is in progress are not guaranteed to be in the list.*

- **void setProperties** (const util::Properties &properties)

*Sets the **Properties** (p. 3216) this **LogManager** (p. 2480) should use to configure its loggers.*

- **const util::Properties & getProperties** () const

*Gets a reference to the Logging **Properties** (p. 3216) used by this logger.*

- **std::string getProperty** (const std::string &name)

*Gets the value of a named property of this **LogManager** (p. 2480).*

- **void addPropertyChangeListener** (PropertyChangeListener \*listener)

*Adds a change listener for **LogManager** (p. 2480) **Properties** (p. 3216), adding the same instance of a change event listener does nothing.*

- **void removePropertyChangeListener** (PropertyChangeListener \*listener)

*Removes a properties change listener from the **LogManager** (p. 2480), if the listener is not found of the param is NULL this method returns silently.*

- **void readConfiguration** () throw ( decaf::io::IOException )

*Reinitialize the logging properties and reread the logging configuration.*

- **void readConfiguration** (decaf::io::InputStream \*stream) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )

*Reinitialize the logging properties and reread the logging configuration from the given stream, which should be in **decaf.util.Properties** (p. 3216) format.*

- **void reset** ()

*Reset the logging configuration.*

## Static Public Member Functions

- **static LogManager & getLogManager** ()

*Get the global **LogManager** (p. 2480) instance.*

## Protected Member Functions

- **LogManager** ()  
*Constructor, hidden to protect against direct instantiation.*
- **LogManager** (const **LogManager** &manager)  
*Copy Constructor.*
- void **operator=** (const **LogManager** &manager)  
*Assignment operator.*

## Friends

- class **decaf::lang::Runtime**

### 6.493.1 Detailed Description

There is a single global **LogManager** (p. 2480) object that is used to maintain a set of shared state about Loggers and log services. This **LogManager** (p. 2480) object:

\* Manages a hierarchical namespace of **Logger** (p. 2461) objects. All named Loggers are stored in this namespace. \* Manages a set of logging control properties. These are simple key-value pairs that can be used by Handlers and other logging objects to configure themselves.

The global **LogManager** (p. 2480) object can be retrieved using **LogManager::getLogManager()** (p. 2485). The **LogManager** (p. 2480) object is created during class initialization and cannot subsequently be changed.

\*\*\*TODO\*\*\* By default, the **LogManager** (p. 2480) reads its initial configuration from a properties file "lib/logging.properties" in the JRE directory. If you edit that property file you can change the default logging configuration for all uses of that JRE.

In addition, the **LogManager** (p. 2480) uses two optional system properties that allow more control over reading the initial configuration:

\* "decaf.logger.config.class" \* "decaf.logger.config.file"

These two properties may be set via the Preferences API, or as command line property definitions to the "java" command, or as system property definitions passed to JNI\_-CreateJavaVM.

If the "java.util.logging.config.class" property is set, then the property value is treated as a class name. The given class will be loaded, an object will be instantiated, and that object's constructor is responsible for reading in the initial configuration. (That object may use other system properties to control its configuration.) The alternate configuration class can use readConfiguration(InputStream) to define properties in the **LogManager** (p. 2480).

If "decaf.util.logging.config.class" property is not set, then the "decaf.util.logging.config.file" system property can be used to specify a properties file (in **decaf.util.Properties** (p. 3216)

format). The initial logging configuration will be read from this file.

If neither of these properties is defined then, as described above, the **LogManager** (p. 2480) will read its initial configuration from a properties file "lib/logging.properties" in the working directory.

The properties for loggers and Handlers will have names starting with the dot-separated name for the handler or logger. \*\*\*TODO\*\*\*

The global logging properties may include:

- \* A property "handlers". This defines a whitespace separated list of class names for handler classes to load and register as handlers on the root **Logger** (p. 2461) (the **Logger** (p. 2461) named ""). Each class name must be for a **Handler** (p. 2042) class which has a default constructor. Note that these Handlers may be created lazily, when they are first used.
- \* A property "<logger>.handlers". This defines a whitespace or comma separated list of class names for handlers classes to load and register as handlers to the specified logger. Each class name must be for a **Handler** (p. 2042) class which has a default constructor. Note that these Handlers may be created lazily, when they are first used.
- \* A property "<logger>.useParentHandlers". This defines a boolean value. By default every logger calls its parent in addition to handling the logging message itself, this often result in messages being handled by the root logger as well. When setting this property to false a **Handler** (p. 2042) needs to be configured for this logger otherwise no logging messages are delivered.
- \* A property "config". This property is intended to allow arbitrary configuration code to be run. The property defines a whitespace separated list of class names. A new instance will be created for each named class. The default constructor of each class may execute arbitrary code to update the logging configuration, such as setting logger levels, adding handlers, adding filters, etc.

Loggers are organized into a naming hierarchy based on their dot separated names. Thus "a.b.c" is a child of "a.b", but "a.b1" and "a.b2" are peers.

All properties whose names end with ".level" are assumed to define log levels for Loggers. Thus "foo.level" defines a log level for the logger called "foo" and (recursively) for any of its children in the naming hierarchy. Log Levels are applied in the order they are defined in the properties file. Thus level settings for child nodes in the tree should come after settings for their parents. The property name ".level" can be used to set the level for the root of the tree.

All methods on the **LogManager** (p. 2480) object are multi-thread safe.

Since

1.0

## 6.493.2 Constructor & Destructor Documentation

**6.493.2.1** virtual decaf::util::logging::LogManager::~~LogManager ( ) [virtual]

**6.493.2.2** decaf::util::logging::LogManager::LogManager ( ) [protected]

Constructor, hidden to protect against direct instantiation.

**6.493.2.3** `decaf::util::logging::LogManager::LogManager ( const LogManager & manager )`  
`[protected]`

Copy Constructor.

#### Parameters

<i>manager</i>	the Manager to copy
----------------	---------------------

### 6.493.3 Member Function Documentation

**6.493.3.1** `bool decaf::util::logging::LogManager::addLogger ( Logger * logger )`  
`throw ( decaf::lang::exceptions::NullPointerException,`  
`decaf::lang::exceptions::IllegalArgumentException )`

Add a named logger.

This does nothing and returns false if a logger with the same name is already registered.

The **Logger** (p. 2461) factory methods call this method to register each newly created **Logger** (p. 2461).

#### Parameters

<i>logger</i>	The new <b>Logger</b> (p. 2461) instance to add to this <b>LogManager</b> (p. 2480).
---------------	--

#### Exceptions

<i>NullPointerException</i>	if logger is NULL.
<i>IllegalArgumentException</i>	if the logger has no name.

**6.493.3.2** `void decaf::util::logging::LogManager::addPropertyChangeListener (`  
`PropertyChangeListener * listener )`

Adds a change listener for **LogManager** (p. 2480) **Properties** (p. 3216), adding the same instance of a change event listener does nothing.

#### Parameters

<i>listener</i>	The PropertyChangeListener to add (can be NULL).
-----------------	--

**6.493.3.3** `Logger* decaf::util::logging::LogManager::getLogger ( const std::string & name )`

Retrieves or creates a new **Logger** (p. 2461) using the name specified a new logger inherits the configuration of the logger's parent if there is no configuration data for the logger.

**Parameters**

<i>name</i>	The name of the <b>Logger</b> (p. 2461).
-------------	--

**6.493.3.4** `int decaf::util::logging::LogManager::getLoggerNames ( const std::vector< std::string > & names )`

Gets a list of known **Logger** (p. 2461) Names from this Manager, new loggers added while this method is in progress are not guaranteed to be in the list.

**Parameters**

<i>names</i>	STL Vector to hold string logger names.
--------------	---

**Returns**

names count of how many loggers were inserted.

**6.493.3.5** `static LogManager& decaf::util::logging::LogManager::getLogManager ( )`  
[static]

Get the global **LogManager** (p. 2480) instance.

**Returns**

A reference to the global **LogManager** (p. 2480) instance.

**6.493.3.6** `const util::Properties& decaf::util::logging::LogManager::getProperties ( ) const`  
[inline]

Gets a reference to the Logging **Properties** (p. 3216) used by this logger.

**Returns**

The **Logger** (p. 2461) **Properties** (p. 3216) Pointer

**6.493.3.7** `std::string decaf::util::logging::LogManager::getProperty ( const std::string & name )`

Gets the value of a named property of this **LogManager** (p. 2480).

**Parameters**

<i>name</i>	The name of the Property to retrieve.
-------------	---------------------------------------

**Returns**

the value of the property

**6.493.3.8** void decaf::util::logging::LogManager::operator= ( const LogManager & *manager* )  
[protected]

Assignment operator.

#### Parameters

<i>manager</i>	the manager to assign from
----------------	----------------------------

**6.493.3.9** void decaf::util::logging::LogManager::readConfiguration ( ) throw ( decaf::io::IOException )

Reinitialize the logging properties and reread the logging configuration.

The same rules are used for locating the configuration properties as are used at startup. So normally the logging properties will be re-read from the same file that was used at startup.

Any log level definitions in the new configuration file will be applied using **Logger.setLevel()** (p. 2472), if the target **Logger** (p. 2461) exists.

A PropertyChangeEvent will be fired after the properties are read.

#### Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

**6.493.3.10** void decaf::util::logging::LogManager::readConfiguration ( decaf::io::InputStream \* *stream* ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )

Reinitialize the logging properties and reread the logging configuration from the given stream, which should be in **decaf.util.Properties** (p. 3216) format.

A PropertyChangeEvent will be fired after the properties are read.

Any log level definitions in the new configuration file will be applied using **Logger.setLevel()** (p. 2472), if the target **Logger** (p. 2461) exists.

#### Parameters

<i>stream</i>	The InputStream to read the <b>Properties</b> (p. 3216) from.
---------------	---

#### Exceptions

<i>NullPointerException</i>	if stream is NULL.
<i>IOException</i>	if an I/O error occurs.



**6.493.3.11** void decaf::util::logging::LogManager::removePropertyChangeListener (   
 PropertyChangeListener \* *listener* )

Removes a properties change listener from the **LogManager** (p. 2480), if the listener is not found of the param is NULL this method returns silently.

#### Parameters

<i>listener</i>	The PropertyChangeListener to remove from the listeners set.
-----------------	--

**6.493.3.12** void decaf::util::logging::LogManager::reset ( )

Reset the logging configuration.

For all named loggers, the reset operation removes and closes all Handlers and (except for the root logger) sets the level to INHERIT. The root logger's level is set to **Level::INFO** (p. 2408).

**6.493.3.13** void decaf::util::logging::LogManager::setProperties ( const util::Properties &   
 *properties* )

Sets the **Properties** (p. 3216) this **LogManager** (p. 2480) should use to configure its loggers.

Once set a properties change event is fired.

#### Parameters

<i>properties</i>	Pointer to read the configuration from
-------------------	--

## 6.493.4 Friends And Related Function Documentation

**6.493.4.1** friend class decaf::lang::Runtime [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/LogManager.h

## 6.494 decaf::util::logging::LogRecord Class Reference

**LogRecord** (p. 2487) objects are used to pass logging requests between the logging framework and individual log Handlers.

```
#include <src/main/decaf/util/logging/LogRecord.h>
```

## Public Member Functions

- **LogRecord ()**
- virtual **~LogRecord ()**
- **Level getLevel ()** const  
*Get **Level** (p. 2403) of this log record.*
- void **setLevel (Level value)**  
*Set (p. 3538) the **Level** (p. 2403) of this Log Record.*
- const std::string & **getLoggerName ()** const  
*Gets the Source Logger's Name.*
- void **setLoggerName** (const std::string &loggerName)  
*Sets the Source Logger's Name.*
- const std::string & **getSourceFile ()** const  
*Gets the Source Log File name.*
- void **setSourceFile** (const std::string &sourceFile)  
*Sets the Source Log File Name.*
- unsigned int **getSourceLine ()** const  
*Gets the Source Log line number.*
- void **setSourceLine** (unsigned int sourceLine)  
*Sets the Source Log line number.*
- const std::string & **getMessage ()** const  
*Gets the Message to be Logged.*
- void **setMessage** (const std::string &message)  
*Sets the Message to be Logged.*
- const std::string & **getSourceFunction ()** const  
*Gets the name of the function where this log was logged.*
- void **setSourceFunction** (const std::string &functionName)  
*Sets the name of the function where this log was logged.*
- long long **getTimestamp ()** const  
*Gets the time in mills that this message was logged.*
- void **setTimestamp** (long long timeStamp)  
*Sets the time in mills that this message was logged.*

- long long **getThreadId** () const  
*Gets the Thread Id where this Log was created.*
- void **setThreadId** (long long threadId)  
*Sets the Thread Id where this Log was created.*
- **decaf::lang::Throwable \* getThrown** () const  
*Gets any Throwable associated with this **LogRecord** (p. 2487).*
- void **setThrown** (**decaf::lang::Throwable** \*thrown)  
*Sets the Throwable associated with this **LogRecord** (p. 2487), the pointer becomes the property of this instance of the **LogRecord** (p. 2487) and will be deleted when the record is destroyed.*

### 6.494.1 Detailed Description

**LogRecord** (p. 2487) objects are used to pass logging requests between the logging framework and individual log Handlers. When a **LogRecord** (p. 2487) is passed into the logging framework it logically belongs to the framework and should no longer be used or updated by the client application.

Since

1.0

### 6.494.2 Constructor & Destructor Documentation

6.494.2.1 **decaf::util::logging::LogRecord::LogRecord** ( )

6.494.2.2 **virtual decaf::util::logging::LogRecord::~~LogRecord** ( ) [virtual]

### 6.494.3 Member Function Documentation

6.494.3.1 **Level decaf::util::logging::LogRecord::getLevel** ( ) const [inline]

Get **Level** (p. 2403) of this log record.

Returns

**Level** (p. 2403) enumeration value.

6.494.3.2 **const std::string& decaf::util::logging::LogRecord::getLoggerName** ( ) const [inline]

Gets the Source Logger's Name.

**Returns**

the source loggers name

**6.494.3.3** `const std::string& decaf::util::logging::LogRecord::getMessage ( ) const`  
[inline]

Gets the Message to be Logged.

**Returns**

the source logger's message

**6.494.3.4** `const std::string& decaf::util::logging::LogRecord::getSourceFile ( ) const`  
[inline]

Gets the Source Log File name.

**Returns**

the source loggers name

**6.494.3.5** `const std::string& decaf::util::logging::LogRecord::getSourceFunction ( ) const`  
[inline]

Gets the name of the function where this log was logged.

**Returns**

the source logger's message

**6.494.3.6** `unsigned int decaf::util::logging::LogRecord::getSourceLine ( ) const` [inline]

Gets the Source Log line number.

**Returns**

the source loggers line number

**6.494.3.7** `decaf::lang::Throwable* decaf::util::logging::LogRecord::getThrown ( ) const`  
[inline]

Gets any Throwable associated with this **LogRecord** (p. 2487).

**Returns**

point to a Throwable instance or Null.

**6.494.3.8** `long long decaf::util::logging::LogRecord::getTimestamp ( ) const` `[inline]`

Gets the time in mills that this message was logged.

#### Returns

UTC time in milliseconds

**6.494.3.9** `long long decaf::util::logging::LogRecord::getTreadId ( ) const` `[inline]`

Gets the Thread Id where this Log was created.

#### Returns

the source loggers line number

**6.494.3.10** `void decaf::util::logging::LogRecord::setLevel ( Level value )` `[inline]`

Set (p. 3538) the **Level** (p. 2403) of this Log Record.

#### Parameters

<i>value</i>	<b>Level</b> (p. 2403) Enumeration Value
--------------	--

**6.494.3.11** `void decaf::util::logging::LogRecord::setLoggerName ( const std::string & loggerName )` `[inline]`

Sets the Source Logger's Name.

#### Parameters

<i>loggerName</i>	the source loggers name
-------------------	-------------------------

**6.494.3.12** `void decaf::util::logging::LogRecord::setMessage ( const std::string & message )` `[inline]`

Sets the Message to be Logged.

#### Parameters

<i>message</i>	the source loggers message
----------------	----------------------------

**6.494.3.13** `void decaf::util::logging::LogRecord::setSourceFile ( const std::string & sourceFile )`  
`[inline]`

Sets the Source Log File Name.

#### Parameters

<i>sourceFile</i>	the source loggers name
-------------------	-------------------------

**6.494.3.14** `void decaf::util::logging::LogRecord::setSourceFunction ( const std::string & functionName )`  
`[inline]`

Sets the name of the function where this log was logged.

#### Parameters

<i>function-Name</i>	the source of the log
----------------------	-----------------------

**6.494.3.15** `void decaf::util::logging::LogRecord::setSourceLine ( unsigned int sourceLine )`  
`[inline]`

Sets the Source Log line number.

#### Parameters

<i>sourceLine</i>	the source logger's line number
-------------------	---------------------------------

**6.494.3.16** `void decaf::util::logging::LogRecord::setThrown ( decaf::lang::Throwable * thrown )`  
`[inline]`

Sets the Throwable associated with this **LogRecord** (p. 2487), the pointer becomes the property of this instance of the **LogRecord** (p. 2487) and will be deleted when the record is destroyed.

#### Parameters

<i>thrown</i>	A pointer to a Throwable that will be associated with this record.
---------------	--

**6.494.3.17** `void decaf::util::logging::LogRecord::setTimestamp ( long long timeStamp )`  
`[inline]`

Sets the time in mills that this message was logged.

**Parameters**

<i>timeStamp</i>	UTC Time in Milliseconds.
------------------	---------------------------

**6.494.3.18** void decaf::util::logging::LogRecord::setThreadId ( long long *threadId* )  
[inline]

Sets the Thread Id where this Log was created.

**Parameters**

<i>threadId</i>	the source logger's line number
-----------------	---------------------------------

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**LogRecord.h**

**6.495 decaf::util::logging::LogWriter Class Reference**

```
#include <src/main/decaf/util/logging/LogWriter.h>
```

**Public Member Functions**

- **LogWriter** ()
- virtual **~LogWriter** ()
- virtual void **log** (const std::string &file, const int line, const std::string &prefix, const std::string &message)  
*Writes a message to the output destination.*
- virtual void **log** (const std::string &message)  
*Writes a message to the output destination.*

**Static Public Member Functions**

- static **LogWriter** & **getInstance** ()  
*Get the singleton instance.*
- static void **returnInstance** ()  
*Returns a Checked out instance of this Writer.*
- static void **destroy** ()  
*Forcefully Delete the Instance of this **LogWriter** (p. 2493) even if there are outstanding references.*

## 6.495.1 Constructor & Destructor Documentation

6.495.1.1 `decaf::util::logging::LogWriter::LogWriter ( )`

6.495.1.2 `virtual decaf::util::logging::LogWriter::~~LogWriter ( )` [virtual]

## 6.495.2 Member Function Documentation

6.495.2.1 `static void decaf::util::logging::LogWriter::destroy ( )` [static]

Forcefully Delete the Instance of this **LogWriter** (p. 2493) even if there are outstanding references.

6.495.2.2 `static LogWriter& decaf::util::logging::LogWriter::getInstance ( )` [static]

Get the singleton instance.

6.495.2.3 `virtual void decaf::util::logging::LogWriter::log ( const std::string & message )`  
[virtual]

Writes a message to the output destination.

### Parameters

<i>message</i>	
----------------	--

6.495.2.4 `virtual void decaf::util::logging::LogWriter::log ( const std::string & file, const int line, const std::string & prefix, const std::string & message )` [virtual]

Writes a message to the output destination.

### Parameters

<i>file</i>	
<i>line</i>	
<i>prefix</i>	
<i>message</i>	

6.495.2.5 `static void decaf::util::logging::LogWriter::returnInstance ( )` [static]

Returns a Checked out instance of this Writer.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/LogWriter.h`



## 6.496 decaf::lang::Long Class Reference

```
#include <src/main/decaf/lang/Long.h>
```

Inheritance diagram for decaf::lang::Long:

### Public Member Functions

- **Long** (long long value)
- **Long** (const std::string &value) throw ( exceptions::NumberFormatException )
- virtual ~**Long** ()
- virtual int **compareTo** (const **Long** &l) const  
*Compares this **Long** (p. 2495) instance with another.*
- bool **equals** (const **Long** &l) const
- virtual bool **operator==** (const **Long** &l) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Long** &l) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual int **compareTo** (const long long &l) const  
*Compares this **Long** (p. 2495) instance with another.*
- bool **equals** (const long long &l) const
- virtual bool **operator==** (const long long &l) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const long long &l) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- std::string **toString** () const
- virtual double **doubleValue** () const  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const  
*Answers the float value which the receiver represents.*
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual short **shortValue** () const

*Answers the short value which the receiver represents.*

- virtual int **intValue** () const

*Answers the int value which the receiver represents.*

- virtual long long **longValue** () const

*Answers the long value which the receiver represents.*

## Static Public Member Functions

- static int **bitCount** (long long value)

*Returns the number of one-bits in the two's complement binary representation of the specified int value.*

- static **Long decode** (const std::string &value) throw ( exceptions::NumberFormatException )

*Decodes a **String** (p. 3775) into a **Long** (p. 2495).*

- static long long **highestOneBit** (long long value)

*Returns an long long value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.*

- static long long **lowestOneBit** (long long value)

*Returns an long long value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.*

- static int **numberOfLeadingZeros** (long long value)

*Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value.*

- static int **numberOfTrailingZeros** (long long value)

*Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value.*

- static long long **parseLong** (const std::string &value) throw ( exceptions::NumberFormatException )

*Parses the string argument as a signed decimal long.*

- static long long **parseLong** (const std::string &value, int radix) throw ( exceptions::NumberFormatException )

*Returns a **Long** (p. 2495) object holding the value extracted from the specified string when parsed with the radix given by the second argument.*

- static long long **reverseBytes** (long long value)

*Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified long long value.*

- static long long **reverse** (long long value)  
*Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified long long value.*
- static long long **rotateLeft** (long long value, int distance)  
*Returns the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits.*
- static long long **rotateRight** (long long value, int distance)  
*Returns the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits.*
- static int **signum** (long long value)  
*Returns the signum function of the specified value.*
- static std::string **toString** (long long value)  
*Converts the long to a **String** (p. 3775) representation.*
- static std::string **toString** (long long value, int radix)
- static std::string **toHexString** (long long value)  
*Returns a string representation of the integer argument as an unsigned integer in base 16.*
- static std::string **toOctalString** (long long value)  
*Returns a string representation of the long long argument as an unsigned long long in base 8.*
- static std::string **toBinaryString** (long long value)  
*Returns a string representation of the long long argument as an unsigned long long in base 2.*
- static **Long valueOf** (long long value)  
*Returns a **Long** (p. 2495) instance representing the specified int value.*
- static **Long valueOf** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Returns a **Long** (p. 2495) object holding the value given by the specified std::string.*
- static **Long valueOf** (const std::string &value, int radix) throw ( exceptions::NumberFormatException )  
*Returns a **Long** (p. 2495) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

## Static Public Attributes

- static const int **SIZE** = 64  
*The size in bits of the primitive long long type.*
- static const long long **MAX\_VALUE** = (long long)0x7FFFFFFFFFFFFFFFLL  
*The maximum value that the primitive type can hold.*
- static const long long **MIN\_VALUE** = (long long)0x8000000000000000LL  
*The minimum value that the primitive type can hold.*

## 6.496.1 Constructor & Destructor Documentation

### 6.496.1.1 decaf::lang::Long::Long ( long long *value* )

#### Parameters

<i>value</i>	- the primitive long long to wrap
--------------	-----------------------------------

### 6.496.1.2 decaf::lang::Long::Long ( const std::string & *value* ) throw ( exceptions::NumberFormatException )

#### Parameters

<i>value</i>	- the long long formatted string to wrap
--------------	--

#### Exceptions

<i>NumberFormatException</i>	
------------------------------	--

### 6.496.1.3 virtual decaf::lang::Long::~~Long ( ) [inline, virtual]

## 6.496.2 Member Function Documentation

### 6.496.2.1 static int decaf::lang::Long::bitCount ( long long *value* ) [static]

Returns the number of one-bits in the two's complement binary representation of the specified int value.

This function is sometimes referred to as the population count.

#### Parameters

<i>value</i>	- the long long to count
--------------	--------------------------

**Returns**

the number of one-bits in the two's complement binary representation of the specified long long value.

**6.496.2.2** `virtual unsigned char decaf::lang::Long::byteValue ( ) const [inline, virtual]`

Answers the byte value which the receiver represents.

**Returns**

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2919).

**6.496.2.3** `virtual int decaf::lang::Long::compareTo ( const long long & / ) const [virtual]`

Compares this **Long** (p. 2495) instance with another.

**Parameters**

<i>l</i>	- the <b>Integer</b> (p. 2143) instance to be compared
----------	--

**Returns**

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< long long > (p. 1249).

**6.496.2.4** `virtual int decaf::lang::Long::compareTo ( const Long & / ) const [virtual]`

Compares this **Long** (p. 2495) instance with another.

**Parameters**

<i>l</i>	- the <b>Long</b> (p. 2495) instance to be compared
----------	---

**Returns**

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< Long > (p. 1249).

**6.496.2.5** `static Long decaf::lang::Long::decode ( const std::string & value ) throw ( exceptions::NumberFormatException ) [static]`

Decodes a **String** (p. 3775) into a **Long** (p. 2495).

Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the Integer.parseInt method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a NumberFormatException will be thrown. The result is negated if first character of the specified **String** (p. 3775) is the minus sign. No whitespace characters are permitted in the string.

#### Parameters

<i>value</i>	- The string to decode
--------------	------------------------

#### Returns

a **Long** (p. 2495) object containing the decoded value

#### Exceptions

<i>NumberFormatException</i>	if the string is not formatted correctly.
------------------------------	---

**6.496.2.6** `virtual double decaf::lang::Long::doubleValue ( ) const [inline, virtual]`

Answers the double value which the receiver represents.

#### Returns

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2919).

**6.496.2.7** `bool decaf::lang::Long::equals ( const Long & l ) const [inline, virtual]`

#### Parameters

<i>l</i>	- the <b>Long</b> (p. 2495) object to compare against.
----------	--

#### Returns

true if the two **Integer** (p. 2143) Objects have the same value.

Implements **decaf::lang::Comparable< Long >** (p. 1250).

**6.496.2.8** `bool decaf::lang::Long::equals ( const long long & l ) const` `[inline, virtual]`

#### Parameters

<i>l</i>	- the <b>Long</b> (p. 2495) object to compare against.
----------	--

#### Returns

true if the two **Integer** (p. 2143) Objects have the same value.

Implements **decaf::lang::Comparable**< **long long** > (p. 1250).

**6.496.2.9** `virtual float decaf::lang::Long::floatValue ( ) const` `[inline, virtual]`

Answers the float value which the receiver represents.

#### Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2920).

**6.496.2.10** `static long long decaf::lang::Long::highestOneBit ( long long value )` `[static]`

Returns an long long value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.

Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

#### Parameters

<i>value</i>	- the long long to be inspected
--------------	---------------------------------

#### Returns

an long long value with a single one-bit, in the position of the highest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

**6.496.2.11** `virtual int decaf::lang::Long::intValue ( ) const` `[inline, virtual]`

Answers the int value which the receiver represents.

#### Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2920).

**6.496.2.12** `virtual long long decaf::lang::Long::longValue ( ) const` `[inline, virtual]`

Answers the long value which the receiver represents.

#### Returns

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2920).

**6.496.2.13** `static long long decaf::lang::Long::lowestOneBit ( long long value )` `[static]`

Returns an long long value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.

Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

#### Parameters

<i>value</i>	- the long long to be inspected
--------------	---------------------------------

#### Returns

an long long value with a single one-bit, in the position of the lowest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

**6.496.2.14** `static int decaf::lang::Long::numberOfLeadingZeros ( long long value )` `[static]`

Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value.

Returns 64 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Note that this method is closely related to the logarithm base 2. For all positive int values  $x$ :

$\ast \text{floor}(\log_2(x)) = 63 - \text{numberOfLeadingZeros}(x) \ast \text{ceil}(\log_2(x)) = 64 - \text{numberOfLeadingZeros}(x - 1)$

#### Parameters

<i>value</i>	- the long long to be inspected
--------------	---------------------------------

#### Returns

the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value, or 64 if the value is equal to zero.



**6.496.2.15** `static int decaf::lang::Long::numberOfTrailingZeros ( long long value )`  
`[static]`

Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value.

Returns 64 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

#### Parameters

<i>value</i>	- the int to be inspected
--------------	---------------------------

#### Returns

the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value, or 64 if the value is equal to zero.

**6.496.2.16** `virtual bool decaf::lang::Long::operator< ( const long long & l ) const`  
`[inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>l</i>	- the value to be compared to this one.
----------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< long long >** (p. 1250).

**6.496.2.17** `virtual bool decaf::lang::Long::operator< ( const Long & l ) const` `[inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>l</i>	- the value to be compared to this one.
----------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Long >** (p. 1250).

**6.496.2.18** `virtual bool decaf::lang::Long::operator==( const Long & l ) const` `[inline, virtual]`

Compares equality between this object and the one passed.

#### Parameters

<i>l</i>	- the value to be compared to this one.
----------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Long >** (p. 1250).

**6.496.2.19** `virtual bool decaf::lang::Long::operator==( const long long & l ) const` `[inline, virtual]`

Compares equality between this object and the one passed.

#### Parameters

<i>l</i>	- the value to be compared to this one.
----------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< long long >** (p. 1250).

**6.496.2.20** `static long long decaf::lang::Long::parseLong ( const std::string & value ) throw ( exceptions::NumberFormatException )` `[static]`

Parses the string argument as a signed decimal long.

The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting long value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseLong(java.lang.String, int)` method.

Note that the characters LL or ULL are not permitted to appear at the end of this string as would normally be permitted in a C++ program.

#### Parameters

<i>value</i>	- <b>String</b> (p. 3775) to parse
--------------	------------------------------------

**Returns**

long long value

**Exceptions**

<i>NumberFormatException</i>	on invalid string value
------------------------------	-------------------------

**6.496.2.21** static long long decaf::lang::Long::parseLong ( const std::string & *value*, int *radix* )  
throw ( exceptions::NumberFormatException ) [static]

Returns a **Long** (p. 2495) object holding the value extracted from the specified string when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed long in the radix specified by the second argument, exactly as if the arguments were given to the parse-Long(std::string, int) method. The result is a **Long** (p. 2495) object that represents the long long value specified by the string.

**Parameters**

<i>value</i>	- <b>String</b> (p. 3775) to parse
<i>radix</i>	- the base encoding of the string

**Returns**

long long value

**Exceptions**

<i>NumberFormatException</i>	on invalid string value
------------------------------	-------------------------

**6.496.2.22** static long long decaf::lang::Long::reverse ( long long *value* ) [static]

Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified long long value.

**Parameters**

<i>value</i>	- the value whose bits are to be reversed
--------------	---

**Returns**

the reversed bits long long.

**6.496.2.23** `static long long decaf::lang::Long::reverseBytes ( long long value )` `[static]`

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified long long value.

#### Parameters

<i>value</i>	- the long long whose bytes we are to reverse
--------------	---

#### Returns

the reversed long long.

**6.496.2.24** `static long long decaf::lang::Long::rotateLeft ( long long value, int distance )`  
`[static]`

Returns the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits.

(Bits shifted out of the left hand, or high-order, side reenter on the right, or low-order.)

Note that left rotation with a negative distance is equivalent to right rotation: `rotateLeft(val, -distance) == rotateRight(val, distance)`. Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: `rotateLeft(val, distance) == rotateLeft(val, distance & 0x1F)`.

#### Parameters

<i>value</i>	- the long long to be inspected
<i>distance</i>	- the number of bits to rotate

#### Returns

the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits.

**6.496.2.25** `static long long decaf::lang::Long::rotateRight ( long long value, int distance )`  
`[static]`

Returns the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits.

(Bits shifted out of the right hand, or low-order, side reenter on the left, or high-order.)

Note that right rotation with a negative distance is equivalent to left rotation: `rotateRight(val, -distance) == rotateLeft(val, distance)`. Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: `rotateRight(val, distance) == rotateRight(val, distance & 0x1F)`.

**Parameters**

<i>value</i>	- the long long to be inspected
<i>distance</i>	- the number of bits to rotate

**Returns**

the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits.

**6.496.2.26** `virtual short decaf::lang::Long::shortValue ( ) const [inline, virtual]`

Answers the short value which the receiver represents.

**Returns**

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2920).

**6.496.2.27** `static int decaf::lang::Long::signum ( long long value ) [static]`

Returns the signum function of the specified value.

(The return value is -1 if the specified value is negative; 0 if the specified value is zero; and 1 if the specified value is positive.)

**Parameters**

<i>value</i>	- the long long to be inspected
--------------	---------------------------------

**Returns**

the signum function of the specified long long value.

**6.496.2.28** `static std::string decaf::lang::Long::toBinaryString ( long long value ) [static]`

Returns a string representation of the long long argument as an unsigned long long in base 2.

The unsigned long long value is the argument plus  $2^{32}$  if the argument is negative; otherwise it is equal to the argument. This value is converted to a string of ASCII digits in binary (base 2) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The characters '0' and '1' are used as binary digits.

**Parameters**

<i>value</i>	- the long long to be translated to a binary string
--------------	---

**Returns**

the unsigned long long value as a binary string

**6.496.2.29 static std::string decaf::lang::Long::toHexString ( long long *value* ) [static]**

Returns a string representation of the integer argument as an unsigned integer in base 16.

The unsigned integer value is the argument plus  $2^{32}$  if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in hexadecimal (base 16) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as hexadecimal digits:

0123456789abcdef

If uppercase letters are desired, the toUpperCase() method may be called on the result:

**Parameters**

<i>value</i>	- the long long to be translated to an Octal string
--------------	---

**Returns**

the unsigned long long value as a Octal string

**6.496.2.30 static std::string decaf::lang::Long::toOctalString ( long long *value* ) [static]**

Returns a string representation of the long long argument as an unsigned long long in base 8.

The unsigned long long value is the argument plus  $2^{32}$  if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in octal (base 8) with no extra leading 0s.

If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as octal digits:

01234567

**Parameters**

<i>value</i>	- the long long to be translated to an Octal string
--------------	---

**Returns**

the unsigned long long value as a Octal string

**6.496.2.31** static std::string decaf::lang::Long::toString ( long long *value* ) [static]

Converts the long to a **String** (p. 3775) representation.

#### Parameters

<i>value</i>	The long to convert to a std::string.
--------------	---------------------------------------

#### Returns

string representation

**6.496.2.32** std::string decaf::lang::Long::toString ( ) const

#### Returns

this **Long** (p. 2495) Object as a **String** (p. 3775) Representation

**6.496.2.33** static std::string decaf::lang::Long::toString ( long long *value*, int *radix* )  
[static]

**6.496.2.34** static Long decaf::lang::Long::valueOf ( long long *value* ) [inline,  
static]

Returns a **Long** (p. 2495) instance representing the specified int value.

#### Parameters

<i>value</i>	- the long long to wrap
--------------	-------------------------

#### Returns

the new **Integer** (p. 2143) object wrapping value.

**6.496.2.35** static Long decaf::lang::Long::valueOf ( const std::string & *value*, int *radix* ) throw  
( exceptions::NumberFormatException ) [static]

Returns a **Long** (p. 2495) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed long long in the radix specified by the second argument, exactly as if the argument were given to the parseLong( std::string, int ) method. The result is a **Long** (p. 2495) object that represents the long long value specified by the string.

#### Parameters

<i>value</i>	- std::string to parse as base ( radix )
<i>radix</i>	- base of the string to parse.

**Returns**

new **Long** (p. 2495) Object wrapping the primitive

**Exceptions**

<i>NumberFormatException</i>	if the string is not a valid long long.
------------------------------	---

**6.496.2.36** `static Long decaf::lang::Long::valueOf ( const std::string & value ) throw ( exceptions::NumberFormatException ) [static]`

Returns a **Long** (p. 2495) object holding the value given by the specified std::string.

The argument is interpreted as representing a signed decimal long long, exactly as if the argument were given to the `parseLong( std::string )` method. The result is a **Integer** (p. 2143) object that represents the long long value specified by the string.

**Parameters**

<i>value</i>	- std::string to parse as base 10
--------------	-----------------------------------

**Returns**

new **Long** (p. 2495) Object wrapping the primitive

**Exceptions**

<i>NumberFormatException</i>	if the string is not a decimal long long.
------------------------------	---

**6.496.3 Field Documentation**

**6.496.3.1** `const long long decaf::lang::Long::MAX_VALUE = (long long)0x7FFFFFFFFFFFFFFFLL [static]`

The maximum value that the primitive type can hold.

**6.496.3.2** `const long long decaf::lang::Long::MIN_VALUE = (long long)0x8000000000000000LL [static]`

The minimum value that the primitive type can hold.

**6.496.3.3** `const int decaf::lang::Long::SIZE = 64 [static]`

The size in bits of the primitive long long type.

The documentation for this class was generated from the following file:



- src/main/decaf/lang/Long.h

## 6.497 decaf::internal::nio::LongArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/LongArrayBuffer.h>
```

Inheritance diagram for decaf::internal::nio::LongArrayBuffer:

### Public Member Functions

- **LongArrayBuffer** (int size, bool readOnly=false) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **IntArrayBuffer** (p. 2119) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **LongArrayBuffer** (long long \*array, int size, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a **LongArrayBuffer** (p. 2511) object that wraps the given array.*
- **LongArrayBuffer** (const decaf::lang::Pointer< ByteArrayAdapter > &array, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.*
- **LongArrayBuffer** (const LongArrayBuffer &other)  
*Create a **LongArrayBuffer** (p. 2511) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*
- virtual ~**LongArrayBuffer** ()
- virtual long long \* **array** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )  
*Returns the long long array that backs this buffer (optional operation).  
Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.  
Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.*  
**Returns**  
*the array that backs this **Buffer** (p. 936).*  
**Exceptions**

<b>ReadOnlyBufferException</b> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
---	--

UnsupportedOperation Exception	if the underlying store has no array.
-----------------------------------	---------------------------------------

- virtual int **arrayOffset** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

The offset long longo the backing array where index zero starts.

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
UnsupportedOperation Exception	if the underlying store has no array.

- virtual LongBuffer \* **asReadOnlyBuffer** () const

Creates a new, read-only long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns

The new, read-only long long buffer which the caller then owns.

- virtual LongBuffer & **compact** () throw ( decaf::nio::ReadOnlyBufferException )

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 941) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 940) - 1 is copied to index  $n = \text{limit}()$  (p. 940) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

#### Returns

a reference to this **LongBuffer** (p. 2522).

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only.
---	------------------------------

- virtual LongBuffer \* **duplicate** ()

*Creates a new long long buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns**

*a new long long **Buffer** (p. 936) which the caller owns.*

- virtual long long **get** () throw ( decaf::nio::BufferUnderflowException )

*Relative get method.*

*Reads the value at this buffer's current position, and then increments the position.*

**Returns**

*the long long at the current position.*

**Exceptions**

<b>BufferUnderflowException</b> (p. 967)	if there no more data to return.
---	----------------------------------

- virtual long long **get** (int index) const throw ( lang::exceptions::IndexOutOfBoundsException )

*Absolute get method.*

*Reads the value at the given index.*

**Parameters**

index	The index in the <b>Buffer</b> (p. 936) where the long long is to be read.
-------	--

**Returns**

*the long long that is located at the given index.*

**Exceptions**

IndexOutOfBoundsException	if index is not smaller than the buffer's limit, or index is negative.
---------------------------	--

- virtual bool **hasArray** () const

*Tells whether or not this buffer is backed by an accessible long long array.*

*If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.*

**Returns**

*true if, and only if, this buffer is backed by an array and is not read-only.*

- virtual bool **isReadOnly** () const

*Tells whether or not this buffer is read-only.*

**Returns**

*true if, and only if, this buffer is read-only.*

- virtual LongBuffer & **put** (long long value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes the given long longs long longo this buffer at the current position, and then increments the position.*

**Parameters**

value	<i>The long longs value to be written.</i>
-------	--

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 964)	<i>if this buffer's current position is not smaller than its limit</i>
<b>ReadOnlyBufferException</b> (p. 3260)	<i>if this buffer is read-only</i>

- virtual LongBuffer & **put** (int index, long long value) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes the given long longs long longo this buffer at the given index.*

**Parameters**

index	<i>The position in the <b>Buffer</b> (p. 936) to write the data</i>
value	<i>The long longs to write.</i>

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>IndexOutOfBoundsException</b>	<i>if index greater than the buffer's limit minus the size of the type being written.</i>
<b>ReadOnlyBufferException</b> (p. 3260)	<i>if this buffer is read-only</i>

- virtual LongBuffer \* **slice** () const

*Creates a new **LongBuffer** (p. 2522) whose content is a shared subsequence of this buffer's content.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of*

bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the newly create **LongBuffer** (p. 2522) which the caller owns.

## Protected Member Functions

- virtual void **setReadOnly** (bool value)

Sets this **LongArrayBuffer** (p. 2511) as Read-Only.

## 6.497.1 Constructor & Destructor Documentation

**6.497.1.1** decaf::internal::nio::LongArrayBuffer::LongArrayBuffer ( int size, bool readOnly = false ) throw ( decaf::lang::exceptions::IllegalArgumentException )

Creates a **IntArrayBuffer** (p. 2119) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

### Parameters

<i>size</i>	The size of the array, this is the limit we read and write to.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

### Exceptions

<i>IllegalArgumentEx-ception</i>	if the capacity value is negative.
----------------------------------	------------------------------------

**6.497.1.2** decaf::internal::nio::LongArrayBuffer::LongArrayBuffer ( long long \* array, int size, int offset, int length, bool readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a **LongArrayBuffer** (p. 2511) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

### Parameters

<i>array</i>	The actual array to wrap.
<i>size</i>	The size of the given array.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

**Exceptions**

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if offset is greater than array capacity.

**6.497.1.3** `decaf::internal::nio::LongArrayBuffer::LongArrayBuffer ( const decaf::lang::Pointer< ByteArrayAdapter > & array, int offset, int length, bool readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte buffer that wraps the passed `ByteArrayAdapter` and start at the given offset.

The capacity and limit of the new **LongArrayBuffer** (p. 2511) will be that of the remaining capacity of the passed buffer.

**Parameters**

<i>array</i>	The <code>ByteArrayAdapter</code> to wrap.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

**Exceptions**

<i>NullPointerException</i>	if array is NULL
<i>IndexOutOfBoundsException</i>	if offset + length is greater than array size.

**6.497.1.4** `decaf::internal::nio::LongArrayBuffer::LongArrayBuffer ( const LongArrayBuffer & other )`

Create a **LongArrayBuffer** (p. 2511) that mirrors this one, meaning it shares a reference to this buffers `ByteArrayAdapter` and when changes are made to that data it is reflected in both.

**Parameters**

<i>other</i>	The <b>LongArrayBuffer</b> (p. 2511) this one is to mirror.
--------------	---

6.497.1.5 virtual decaf::internal::nio::LongArrayBuffer::~~LongArrayBuffer ( ) [virtual]

## 6.497.2 Member Function Documentation

6.497.2.1 virtual long long\* decaf::internal::nio::LongArrayBuffer::array ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException ) [virtual]

Returns the long long array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

the array that backs this **Buffer** (p. 936).

### Exceptions

<i>ReadOnlyBufferException</i> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implements decaf::nio::LongBuffer (p. 2526).

6.497.2.2 virtual int decaf::internal::nio::LongArrayBuffer::arrayOffset ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException ) [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

The offset long longo the backing array where index zero starts.

### Exceptions

<i>ReadOnlyBufferException</i> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implements **decaf::nio::LongBuffer** (p. 2526).

**6.497.2.3** `virtual LongBuffer* decaf::internal::nio::LongArrayBuffer::asReadOnlyBuffer ( )`  
`const [virtual]`

Creates a new, read-only long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns

The new, read-only long long buffer which the caller then owns.

Implements **decaf::nio::LongBuffer** (p. 2527).

**6.497.2.4** `virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::compact ( ) throw (`  
`decaf::nio::ReadOnlyBufferException) [virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 941) is copied to index zero, the byte at index `p + 1` is copied to index one, and so forth until the byte at index `limit()` (p. 940) - 1 is copied to index `n = limit()` (p. 940) - 1 - `p`. The buffer's position is then set to `n+1` and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

#### Returns

a reference to this **LongBuffer** (p. 2522).

#### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.
--	------------------------------

Implements **decaf::nio::LongBuffer** (p. 2527).



**6.497.2.5** virtual LongBuffer\* decaf::internal::nio::LongArrayBuffer::duplicate ( )  
[virtual]

Creates a new long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

a new long long **Buffer** (p. 936) which the caller owns.

Implements **decaf::nio::LongBuffer** (p. 2528).

**6.497.2.6** virtual long long decaf::internal::nio::LongArrayBuffer::get ( ) throw ( decaf::nio::BufferUnderflowException ) [virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

#### Returns

the long long at the current position.

#### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 967)	if there no more data to return.
--	----------------------------------

Implements **decaf::nio::LongBuffer** (p. 2530).

**6.497.2.7** virtual long long decaf::internal::nio::LongArrayBuffer::get ( int *index* ) const throw ( lang::exceptions::IndexOutOfBoundsException ) [virtual]

Absolute get method.

Reads the value at the given index.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the long long is to be read.
--------------	--

#### Returns

the long long that is located at the given index.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implements **decaf::nio::LongBuffer** (p. 2528).

**6.497.2.8** `virtual bool decaf::internal::nio::LongArrayBuffer::hasArray ( ) const` [inline, virtual]

Tells whether or not this buffer is backed by an accessible long long array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns**

true if, and only if, this buffer is backed by an array and is not read-only.

Implements **decaf::nio::LongBuffer** (p. 2530).

**6.497.2.9** `virtual bool decaf::internal::nio::LongArrayBuffer::isReadOnly ( ) const` [inline, virtual]

Tells whether or not this buffer is read-only.

**Returns**

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 940).

**6.497.2.10** `virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::put ( int index, long long value ) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )` [virtual]

Writes the given long longs long longo this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data
<i>value</i>	The long longs to write.

**Returns**

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written.
<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only

Implements **decaf::nio::LongBuffer** (p. 2531).

**6.497.2.11** virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::put ( long long *value* ) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException ) [virtual]

Writes the given long longs long longo this buffer at the current position, and then increments the position.

**Parameters**

<i>value</i>	The long longs value to be written.
--------------	-------------------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b>BufferOverflowException</b> (p. 964)	if this buffer's current position is not smaller than its limit
<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only

Implements **decaf::nio::LongBuffer** (p. 2530).

**6.497.2.12** virtual void decaf::internal::nio::LongArrayBuffer::setReadOnly ( bool *value* ) [inline, protected, virtual]

Sets this **LongArrayBuffer** (p. 2511) as Read-Only.

**Parameters**

<i>value</i>	Boolean value, true if this buffer is to be read-only, false otherwise.
--------------	---

**6.497.2.13** `virtual LongBuffer* decaf::internal::nio::LongArrayBuffer::slice ( ) const`  
`[virtual]`

Creates a new **LongBuffer** (p.2522) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the newly create **LongBuffer** (p. 2522) which the caller owns.

Implements **decaf::nio::LongBuffer** (p. 2533).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/LongArrayBuffer.h`

## 6.498 decaf::nio::LongBuffer Class Reference

This class defines four categories of operations upon long long buffers:

```
#include <src/main/decaf/nio/LongBuffer.h>
```

Inheritance diagram for `decaf::nio::LongBuffer`:

### Public Member Functions

- `virtual ~LongBuffer ()`
- `virtual std::string toString () const`
- `virtual long long * array ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )`  
*Returns the long long array that backs this buffer (optional operation).*
- `virtual int arrayOffset ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )`  
*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*
- `virtual LongBuffer * asReadOnlyBuffer () const =0`  
*Creates a new, read-only long long buffer that shares this buffer's content.*

- virtual **LongBuffer** & **compact** ()=0 throw ( ReadOnlyBufferException )  
*Compacts this buffer.*
- virtual **LongBuffer** \* **duplicate** ()=0  
*Creates a new long long buffer that shares this buffer's content.*
- virtual long long **get** ()=0 throw ( BufferUnderflowException )  
*Relative get method.*
- virtual long long **get** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Absolute get method.*
- **LongBuffer** & **get** (std::vector< long long > buffer) throw ( BufferUnderflowException )  
*Relative bulk get method.*
- **LongBuffer** & **get** (long long \*buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Relative bulk get method.*
- virtual bool **hasArray** () const =0  
*Tells whether or not this buffer is backed by an accessible long long array.*
- **LongBuffer** & **put** (**LongBuffer** &src) throw ( BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException )  
*This method transfers the long longs remaining in the given source buffer long longo this buffer.*
- **LongBuffer** & **put** (const long long \*buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*This method transfers long longs long longo this buffer from the given source array.*
- **LongBuffer** & **put** (std::vector< long long > &buffer) throw ( BufferOverflowException, ReadOnlyBufferException )  
*This method transfers the entire content of the given source long longs array long longo this buffer.*
- virtual **LongBuffer** & **put** (long long value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )  
*Writes the given long longs long longo this buffer at the current position, and then increments the position.*
- virtual **LongBuffer** & **put** (int index, long long value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )

*Writes the given long longs long longo this buffer at the given index.*

- virtual **LongBuffer** \* **slice** () const =0  
*Creates a new **LongBuffer** (p.2522) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (const **LongBuffer** &value) const
- virtual bool **equals** (const **LongBuffer** &value) const
- virtual bool **operator==** (const **LongBuffer** &value) const
- virtual bool **operator<** (const **LongBuffer** &value) const

### Static Public Member Functions

- static **LongBuffer** \* **allocate** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Allocates a new Double buffer.*
- static **LongBuffer** \* **wrap** (long long \*array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Wraps the passed buffer with a new **LongBuffer** (p. 2522).*
- static **LongBuffer** \* **wrap** (std::vector< long long > &buffer)  
*Wraps the passed STL long long Vector in a **LongBuffer** (p. 2522).*

### Protected Member Functions

- **LongBuffer** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **LongBuffer** (p.2522) object that has its backing array allocated long longernally and is then owned and deleted when this object is deleted.*

### 6.498.1 Detailed Description

This class defines four categories of operations upon long long buffers: o Absolute and relative get and put methods that read and write single long longs; o Relative bulk get

methods that transfer contiguous sequences of long longs from this buffer long longo an array; and o Relative bulk put methods that transfer contiguous sequences of long longs from a long long array or some other long long buffer long longo this buffer o Methods for compacting, duplicating, and slicing a long long buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing long long array long longo a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

## 6.498.2 Constructor & Destructor Documentation

**6.498.2.1** `decaf::nio::LongBuffer::LongBuffer ( int capacity ) throw ( decaf::lang::exceptions::IllegalArgumentException )` `[protected]`

Creates a **LongBuffer** (p. 2522) object that has its backing array allocated long longer-nally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

### Parameters

<i>capacity</i>	The size and limit of the <b>Buffer</b> (p. 936) in long longs.
-----------------	---

### Exceptions

<i>IllegalArgumentEx-ception</i>	if capacity is negative.
----------------------------------	--------------------------

**6.498.2.2** `virtual decaf::nio::LongBuffer::~~LongBuffer ( )` `[inline, virtual]`

## 6.498.3 Member Function Documentation

**6.498.3.1** `static LongBuffer* decaf::nio::LongBuffer::allocate ( int capacity ) throw ( decaf::lang::exceptions::IllegalArgumentException )` `[static]`

Allocates a new Double buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

### Parameters

<i>capacity</i>	The size of the Double buffer in long longs.
-----------------	--

### Returns

the **LongBuffer** (p. 2522) that was allocated, caller owns.

**6.498.3.2** `virtual long long* decaf::nio::LongBuffer::array ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the long long array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

the array that backs this **Buffer** (p. 936).

### Exceptions

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<i><b>UnsupportedOperationException</b></i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2517).

**6.498.3.3** `virtual int decaf::nio::LongBuffer::arrayOffset ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

The offset long onto the backing array where index zero starts.

### Exceptions

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<i><b>UnsupportedOperationException</b></i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2517).



**6.498.3.4** `virtual LongBuffer* decaf::nio::LongBuffer::asReadOnlyBuffer ( ) const` [pure virtual]

Creates a new, read-only long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns

The new, read-only long long buffer which the caller then owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2518).

**6.498.3.5** `virtual LongBuffer& decaf::nio::LongBuffer::compact ( ) throw ( ReadOnlyBufferException )` [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 941) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 940) - 1 is copied to index  $n = \text{limit}()$  (p. 940) - 1 -  $p$ . The buffer's position is then set to  $n + 1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

#### Returns

a reference to this **LongBuffer** (p. 2522).

#### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.
--	------------------------------

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2518).

**6.498.3.6** `virtual int decaf::nio::LongBuffer::compareTo ( const LongBuffer & value ) const`  
[virtual]

**6.498.3.7** `virtual LongBuffer* decaf::nio::LongBuffer::duplicate ( )` [pure virtual]

Creates a new long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

a new long long **Buffer** (p. 936) which the caller owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2519).

**6.498.3.8** `virtual bool decaf::nio::LongBuffer::equals ( const LongBuffer & value ) const`  
[virtual]

**6.498.3.9** `LongBuffer& decaf::nio::LongBuffer::get ( std::vector< long long > buffer ) throw ( BufferUnderflowException )`

Relative bulk get method.

This method transfers values from this buffer long long the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize( N )` before calling this get method.

### Returns

a reference to this **Buffer** (p. 936).

### Exceptions

<i><b>BufferUnderflowException</b></i> (p. 967)	if there are fewer than length long longs remaining in this buffer.
--	---

**6.498.3.10** `virtual long long decaf::nio::LongBuffer::get ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [pure virtual]

Absolute get method.

Reads the value at the given index.

**Parameters**

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the long long is to be read.
--------------	--

**Returns**

the long long that is located at the given index.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2519).

**6.498.3.11 LongBuffer& decaf::nio::LongBuffer::get ( long long \* *buffer*, int *size*, int *offset*, int *length* ) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )**

Relative bulk get method.

This method transfers long longs from this buffer long longo the given destination array. If there are fewer long longs remaining in the buffer than are required to satisfy the request, that is, if *length* > **remaining()** (p. 941), then no bytes are transferred and a **BufferUnderflowException** (p. 967) is thrown.

Otherwise, this method copies *length* long longs from this buffer long longo the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by *length*.

**Parameters**

<i>buffer</i>	The pointer to an allocated long long buffer to fill.
<i>size</i>	The size of the passed in buffer.
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The amount of data to put in the passed buffer.

**Returns**

a reference to this **Buffer** (p. 936).

**Exceptions**

<i>BufferUnderflowException</i> (p. 967)	if there are fewer than <i>length</i> long longs remaining in this buffer
<i>NullPointerException</i>	longerException if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of <i>size</i> , <i>offset</i> , or <i>length</i> are not met.

**6.498.3.12** `virtual long long decaf::nio::LongBuffer::get ( ) throw ( BufferUnderflowException )` [pure virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

#### Returns

the long long at the current position.

#### Exceptions

<i>BufferUnderflowException</i> (p. 967)	if there no more data to return.
---	----------------------------------

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2519).

**6.498.3.13** `virtual bool decaf::nio::LongBuffer::hasArray ( ) const` [pure virtual]

Tells whether or not this buffer is backed by an accessible long long array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

#### Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2520).

**6.498.3.14** `virtual bool decaf::nio::LongBuffer::operator< ( const LongBuffer & value ) const` [virtual]

**6.498.3.15** `virtual bool decaf::nio::LongBuffer::operator== ( const LongBuffer & value ) const` [virtual]

**6.498.3.16** `virtual LongBuffer& decaf::nio::LongBuffer::put ( long long value ) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes the given long longs long longo this buffer at the current position, and then increments the position.

#### Parameters

<i>value</i>	The long longs value to be written.
--------------	-------------------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 964)	if this buffer's current position is not smaller than its limit
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2521).

**6.498.3.17** virtual **LongBuffer&** decaf::nio::LongBuffer::put ( int *index*, long long *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException ) [pure virtual]

Writes the given long longs long longo this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data
<i>value</i>	The long longs to write.

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>IndexOutOfBoundsException</i></b>	if index greater than the buffer's limit minus the size of the type being written.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2520).

**6.498.3.18** **LongBuffer&** decaf::nio::LongBuffer::put ( const long long \* *buffer*, int *size*, int *offset*, int *length* ) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

This method transfers long longs long longo this buffer from the given source array.

If there are more long longs to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 941), then no long longs are transferred and a **BufferOverflowException** (p. 964) is thrown.

Otherwise, this method copies length bytes from the given array long longo this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by length.

#### Parameters

<i>buffer</i>	The array from which long longs are to be read.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The offset within the array of the first char to be read.
<i>length</i>	The number of long longs to be read from the given array.

#### Returns

a reference to this buffer.

#### Exceptions

<b><i>BufferOverflowException</i></b> (p. 964)	if there is insufficient space in this buffer
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only
<i>NullPolong</i>	longerException if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

#### 6.498.3.19 LongBuffer& decaf::nio::LongBuffer::put ( std::vector< long long > & buffer ) throw ( BufferOverflowException, ReadOnlyBufferException )

This method transfers the entire content of the given source long longs array long longo this buffer.

This is the same as calling put( &buffer[0], 0, buffer.size()).

#### Parameters

<i>buffer</i>	The buffer whose contents are copied to this <b>LongBuffer</b> (p. 2522).
---------------	---

#### Returns

a reference to this buffer.

#### Exceptions

<b><i>BufferOverflowException</i></b> (p. 964)	if there is insufficient space in this buffer.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.

### 6.498.3.20 LongBuffer& decaf::nio::LongBuffer::put ( LongBuffer & src ) throw ( BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException )

This method transfers the long longs remaining in the given source buffer long longo this buffer.

If there are more long longs remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 941), then no long longs are transferred and a **BufferOverflowException** (p. 964) is thrown.

Otherwise, this method copies `n = src.remaining()` long longs from the given buffer long longo this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

#### Parameters

<code>src</code>	The buffer to take long longs from an place in this one.
------------------	--

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 964)	if there is insufficient space in this buffer for the remaining long longs in the source buffer
<b>IllegalArgumentException</b>	if the source buffer is this buffer
<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only

### 6.498.3.21 virtual LongBuffer\* decaf::nio::LongBuffer::slice ( ) const [pure virtual]

Creates a new **LongBuffer** (p. 2522) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the newly create **LongBuffer** (p. 2522) which the caller owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2522).

**6.498.3.22** `virtual std::string decaf::nio::LongBuffer::toString ( ) const` [virtual]

#### Returns

a std::string describing this object

**6.498.3.23** `static LongBuffer* decaf::nio::LongBuffer::wrap ( long long * array, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [static]

Wraps the passed buffer with a new **LongBuffer** (p. 2522).

The new buffer will be backed by the given long long array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>array</i>	The array that will back the new buffer.
<i>size</i>	The size of the passed in array.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

#### Returns

a new **LongBuffer** (p. 2522) that is backed by buffer, caller owns.

#### Exceptions

<i>NullPointerException</i>	if the array pointer is NULL.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

**6.498.3.24** `static LongBuffer* decaf::nio::LongBuffer::wrap ( std::vector< long long > & buffer )` [static]

Wraps the passed STL long long Vector in a **LongBuffer** (p. 2522).

The new buffer will be backed by the given long long array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).
---------------	--



### Returns

a new **LongBuffer** (p. 2522) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**LongBuffer.h**

## 6.499 activemq::util::LongSequenceGenerator Class Reference

This class is used to generate a sequence of long long values that are incremented each time a new value is requested.

```
#include <src/main/activemq/util/LongSequenceGenerator.h>
```

### Public Member Functions

- **LongSequenceGenerator** ()
- virtual **~LongSequenceGenerator** ()
- long long **getNextSequenceId** ()
- long long **getLastSequenceId** ()

#### 6.499.1 Detailed Description

This class is used to generate a sequence of long long values that are incremented each time a new value is requested. This class is thread safe so the ids can be requested in different threads safely.

#### 6.499.2 Constructor & Destructor Documentation

6.499.2.1 **activemq::util::LongSequenceGenerator::LongSequenceGenerator** ( )

6.499.2.2 **virtual activemq::util::LongSequenceGenerator::~~LongSequenceGenerator** ( )  
[inline, virtual]

#### 6.499.3 Member Function Documentation

6.499.3.1 **long long activemq::util::LongSequenceGenerator::getLastSequenceId** ( )

### Returns

the last id that was generated.

### 6.499.3.2 long long activemq::util::LongSequenceGenerator::getNextSequenceId ( )

#### Returns

the next id in the sequence.

The documentation for this class was generated from the following file:

- src/main/activemq/util/**LongSequenceGenerator.h**

## 6.500 decaf::net::MalformedURLException Class Reference

```
#include <src/main/decaf/net/MalformedURLException.h>
```

Inheritance diagram for decaf::net::MalformedURLException:

### Public Member Functions

- **MalformedURLException** () throw ()  
*Default Constructor.*
- **MalformedURLException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **MalformedURLException** (const **MalformedURLException** &ex) throw ()  
*Copy Constructor.*
- **MalformedURLException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **MalformedURLException** (const std::exception \*cause) throw ()  
*Constructor.*
- **MalformedURLException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **MalformedURLException** \* clone () const  
*Clones this exception.*
- virtual ~**MalformedURLException** () throw ()

## 6.500.1 Constructor & Destructor Documentation

**6.500.1.1** `decaf::net::MalformedURLException::MalformedURLException ( ) throw ()`  
`[inline]`

Default Constructor.

**6.500.1.2** `decaf::net::MalformedURLException::MalformedURLException ( const Exception & ex ) throw ()` `[inline]`

Conversion Constructor from some other Exception.

### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.500.1.3** `decaf::net::MalformedURLException::MalformedURLException ( const MalformedURLException & ex ) throw ()` `[inline]`

Copy Constructor.

### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.500.1.4** `decaf::net::MalformedURLException::MalformedURLException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()`  
`[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.500.1.5** `decaf::net::MalformedURLException::MalformedURLException ( const std::exception * cause ) throw ()` `[inline]`

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.500.1.6** `decaf::net::MalformedURLException::MalformedURLException ( const char * file,  
const int lineNumber, const char * msg, ... ) throw ()` [*inline*]

Constructor - Initializes the file name and line number where this message occurred.  
Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.500.1.7** `virtual decaf::net::MalformedURLException::~~MalformedURLException ( ) throw ()`  
[*inline*, *virtual*]

**6.500.2 Member Function Documentation**

**6.500.2.1** `virtual MalformedURLException* decaf::net::MalformedURLException::clone ( ) const` [*inline*, *virtual*]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns**

a new Exception instance that is a copy of this Exception object.

Reimplemented from `decaf::io::IOException` (p. 2212).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/MalformedURLException.h`

**6.501 decaf::util::Map< K, V, COMPARATOR > Class Template Reference**

**Map** (p. 2538) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

## 6.501 decaf::util::Map< K, V, COMPARATOR > Class Template Reference2541

```
#include <src/main/decaf/util/Map.h>
```

Inheritance diagram for decaf::util::Map< K, V, COMPARATOR >:

### Data Structures

- class **Entry**

### Public Member Functions

- **Map** ()  
*Default constructor - does nothing.*
- virtual **~Map** ()
- virtual bool **equals** (const **Map** &source) const =0  
*Comparison, equality is dependent on the method of determining if the element are equal.*
- virtual void **copy** (const **Map** &source)=0  
*Copies the content of the source map into this map.*
- virtual void **clear** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException )  
*Removes all keys and values from this map.*
- virtual bool **containsKey** (const K &key) const =0  
*Indicates whether or this map contains a value for the given key.*
- virtual bool **containsValue** (const V &value) const =0  
*Indicates whether or this map contains a value for the given value, i.e.*
- virtual bool **isEmpty** () const =0
- virtual std::size\_t **size** () const =0
- virtual V & **get** (const K &key)=0 throw ( lang::exceptions::NoSuchElementException )  
*Gets the value mapped to the specified key in the **Map** (p. 2538).*
- virtual const V & **get** (const K &key) const =0 throw ( lang::exceptions::NoSuchElementException )  
*Gets the value mapped to the specified key in the **Map** (p. 2538).*
- virtual void **put** (const K &key, const V &value)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException )  
*Sets the value for the specified key.*

- virtual void **putAll** (const **Map**< K, V, COMPARATOR > &other)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException )  
*Stores a copy of the Mappings contained in the other **Map** (p. 2538) in this one.*
- virtual V **remove** (const K &key)=0 throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )  
*Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.*
- virtual std::vector< K > **keySet** () const =0  
*Returns a **Set** (p. 3538) view of the mappings contained in this map.*
- virtual std::vector< V > **values** () const =0

### 6.501.1 Detailed Description

template<typename K, typename V, typename COMPARATOR = std::less<K>> class decaf::util::Map< K, V, COMPARATOR >

**Map** (p. 2538) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

### 6.501.2 Constructor & Destructor Documentation

6.501.2.1 template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::Map< K, V, COMPARATOR >::Map ( ) [inline]

Default constructor - does nothing.

6.501.2.2 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual decaf::util::Map< K, V, COMPARATOR >::~~Map ( ) [inline, virtual]

### 6.501.3 Member Function Documentation

6.501.3.1 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR >::clear ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException ) [pure virtual]

Removes all keys and values from this map.

### Exceptions

<i>UnsupportedOperationException</i>	if this map is unmodifiable.
--------------------------------------	------------------------------

## 6.501 decaf::util::Map< K, V, COMPARATOR > Class Template Reference 2543

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1270), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3713), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1270), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1270), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1270), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1270), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1270), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1270), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3713), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3713), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3713), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3713), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3713), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3713), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3713), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3713), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3713), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3713), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3713), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3713), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3713).

```
6.501.3.2  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::Map< K, V, COMPARATOR >::containsKey ( const K &
            key ) const    [pure virtual]
```

Indicates whether or this map contains a value for the given key.

### Parameters

<i>key</i>	The key to look up.
------------	---------------------

### Returns

true if this map contains the value, otherwise false.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1271), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3714), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1271), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1271), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1271), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1271), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1271), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1271), `decaf::util::StlMap<`

`cms::Session *`, `SessionResolver *` > (p. 3714), `decaf::util::StdMap< std::string, WireFormatFactory * >` (p. 3714), `decaf::util::StdMap< std::string, Primitive-ValueNode >` (p. 3714), `decaf::util::StdMap< std::string, cms::Queue * >` (p. 3714), `decaf::util::StdMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3714), `decaf::util::StdMap< std::string, CachedConsumer * >` (p. 3714), `decaf::util::StdMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3714), `decaf::util::StdMap< std::string, TransportFactory * >` (p. 3714), `decaf::util::StdMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3714), `decaf::util::StdMap< int, Pointer< Command > >` (p. 3714), `decaf::util::StdMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3714), `decaf::util::StdMap< std::string, CachedProducer * >` (p. 3714), and `decaf::util::StdMap< std::string, cms::Topic * >` (p. 3714).

**6.501.3.3** `template<typename K, typename V, typename COMPARATOR = std::less<K>>`  
`virtual bool decaf::util::Map< K, V, COMPARATOR >::containsValue ( const V &`  
`value ) const` [pure virtual]

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

#### Parameters

<i>value</i>	The Value to look up.
--------------	-----------------------

#### Returns

true if this map contains the value, otherwise false.

Implemented in `decaf::util::concurrent::ConcurrentStdMap< K, V, COMPARATOR >` (p. 1271), `decaf::util::StdMap< K, V, COMPARATOR >` (p. 3714), `decaf::util::concurrent::ConcurrentStdMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1271), `decaf::util::concurrent::ConcurrentStdMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1271), `decaf::util::concurrent::ConcurrentStdMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1271), `decaf::util::concurrent::ConcurrentStdMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1271), `decaf::util::concurrent::ConcurrentStdMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1271), `decaf::util::concurrent::ConcurrentStdMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1271), `decaf::util::StdMap< cms::Session *, SessionResolver * >` (p. 3714), `decaf::util::StdMap< std::string, WireFormatFactory * >` (p. 3714), `decaf::util::StdMap< std::string, Primitive-ValueNode >` (p. 3714), `decaf::util::StdMap< std::string, cms::Queue * >` (p. 3714), `decaf::util::StdMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3714), `decaf::util::StdMap< std::string, CachedConsumer * >` (p. 3714), `decaf::util::StdMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3714), `decaf::util::StdMap< std::string, TransportFactory * >` (p. 3714), `decaf::util::StdMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR`



## 6.501 decaf::util::Map< K, V, COMPARATOR > Class Template Reference 2545

> (p. 3714), **decaf::util::StdMap< int, Pointer< Command > >** (p. 3714), **decaf::util::StdMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR >** (p. 3714), **decaf::util::StdMap< std::string, CachedProducer \* >** (p. 3714), and **decaf::util::StdMap< std::string, cms::Topic \* >** (p. 3714).

**6.501.3.4** `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual void decaf::util::Map< K, V, COMPARATOR >::copy ( const Map< K, V,  
COMPARATOR > & source ) [pure virtual]`

Copies the content of the source map into this map.

Erases all existing data in this map.

### Parameters

<i>source</i>	The source object to copy from.
---------------	---------------------------------

Implemented in **decaf::util::concurrent::ConcurrentStdMap< K, V, COMPARATOR >** (p. 1272), and **decaf::util::StdMap< K, V, COMPARATOR >** (p. 3714).

**6.501.3.5** `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual bool decaf::util::Map< K, V, COMPARATOR >::equals ( const Map< K, V,  
COMPARATOR > & source ) const [pure virtual]`

Comparison, equality is dependent on the method of determining if the element are equal.

### Parameters

<i>source</i>	- <b>Map</b> (p. 2538) to compare to this one.
---------------	--

### Returns

true if the **Map** (p. 2538) passed is equal in value to this one.

Implemented in **decaf::util::concurrent::ConcurrentStdMap< K, V, COMPARATOR >** (p. 1272), and **decaf::util::StdMap< K, V, COMPARATOR >** (p. 3715).

**6.501.3.6** `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual V& decaf::util::Map< K, V, COMPARATOR >::get ( const K & key ) throw (  
lang::exceptions::NoSuchElementException ) [pure virtual]`

Gets the value mapped to the specified key in the **Map** (p. 2538).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** is thrown.

### Parameters

<i>key</i>	The search key.
------------	-----------------

**Returns**

A reference to the value for the given key.

**Exceptions**

<i>NoSuchElementException</i>	if the key requests doesn't exist in the <b>Map</b> (p. 2538).
-------------------------------	--

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1272), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3715), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1272), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1272), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1272), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1272), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 1272), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1272), **decaf::util::StlMap< cms::Session \*, SessionResolver \* >** (p. 3715), **decaf::util::StlMap< std::string, WireFormatFactory \* >** (p. 3715), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 3715), **decaf::util::StlMap< std::string, cms::Queue \* >** (p. 3715), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR >** (p. 3715), **decaf::util::StlMap< std::string, CachedConsumer \* >** (p. 3715), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR >** (p. 3715), **decaf::util::StlMap< std::string, TransportFactory \* >** (p. 3715), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 3715), **decaf::util::StlMap< int, Pointer< Command > >** (p. 3715), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR >** (p. 3715), **decaf::util::StlMap< std::string, CachedProducer \* >** (p. 3715), and **decaf::util::StlMap< std::string, cms::Topic \* >** (p. 3715).

Referenced by **decaf::util::StlMap< std::string, cms::Topic \* >::equals()**, and **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals()**.

```
6.501.3.7  template<typename K, typename V, typename COMPARATOR = std::less<K>>
           virtual const V& decaf::util::Map< K, V, COMPARATOR >::get ( const K & key
           ) const throw ( lang::exceptions::NoSuchElementException ) [pure
           virtual]
```

Gets the value mapped to the specified key in the **Map** (p. 2538).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** is thrown.

**Parameters**

<i>key</i>	The search key.
------------	-----------------

## 6.501 decaf::util::Map< K, V, COMPARATOR > Class Template Reference 2547

### Returns

A {const} reference to the value for the given key.

### Exceptions

<i>NoSuchElementException</i>	if the key requests doesn't exist in the <b>Map</b> (p. 2538).
-------------------------------	--

Implemented in **decaf::util::concurrent::ConcurrentStdMap< K, V, COMPARATOR >** (p. 1273), **decaf::util::StdMap< K, V, COMPARATOR >** (p. 3716), **decaf::util::concurrent::ConcurrentStdMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1273), **decaf::util::concurrent::ConcurrentStdMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1273), **decaf::util::concurrent::ConcurrentStdMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1273), **decaf::util::concurrent::ConcurrentStdMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1273), **decaf::util::concurrent::ConcurrentStdMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 1273), **decaf::util::concurrent::ConcurrentStdMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1273), **decaf::util::StdMap< cms::Session \*, SessionResolver \* >** (p. 3716), **decaf::util::StdMap< std::string, WireFormatFactory \* >** (p. 3716), **decaf::util::StdMap< std::string, PrimitiveValueNode >** (p. 3716), **decaf::util::StdMap< std::string, cms::Queue \* >** (p. 3716), **decaf::util::StdMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR >** (p. 3716), **decaf::util::StdMap< std::string, CachedConsumer \* >** (p. 3716), **decaf::util::StdMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR >** (p. 3716), **decaf::util::StdMap< std::string, TransportFactory \* >** (p. 3716), **decaf::util::StdMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 3716), **decaf::util::StdMap< int, Pointer< Command > >** (p. 3716), **decaf::util::StdMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR >** (p. 3716), **decaf::util::StdMap< std::string, CachedProducer \* >** (p. 3716), and **decaf::util::StdMap< std::string, cms::Topic \* >** (p. 3716).

```
6.501.3.8  template<typename K, typename V, typename COMPARATOR = std::less<K>>
           virtual bool decaf::util::Map< K, V, COMPARATOR >::isEmpty ( ) const
           [pure virtual]
```

### Returns

if the **Map** (p. 2538) contains any element or not, TRUE or FALSE

Implemented in **decaf::util::concurrent::ConcurrentStdMap< K, V, COMPARATOR >** (p. 1273), **decaf::util::StdMap< K, V, COMPARATOR >** (p. 3716), **decaf::util::concurrent::ConcurrentStdMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1273), **decaf::util::concurrent::ConcurrentStdMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1273), **decaf::util::concurrent::ConcurrentStdMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1273), **decaf::util::concurrent::ConcurrentStdMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1273), **decaf::util::concurrent::ConcurrentStdMap<**

**Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR**  
**>** (p. 1273), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId**  
**>, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1273), **decaf::util::StlMap<**  
**cms::Session \*, SessionResolver \* >** (p. 3716), **decaf::util::StlMap< std::string,**  
**WireFormatFactory \* >** (p. 3716), **decaf::util::StlMap< std::string, Primitive-**  
**ValueNode >** (p. 3716), **decaf::util::StlMap< std::string, cms::Queue \* >** (p. 3716),  
**decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*,**  
**commands::ProducerId::COMPARATOR >** (p. 3716), **decaf::util::StlMap< std::string,**  
**CachedConsumer \* >** (p. 3716), **decaf::util::StlMap< Pointer< commands::ConsumerId**  
**>, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR >** (p. 3716),  
**decaf::util::StlMap< std::string, TransportFactory \* >** (p. 3716), **decaf::util::StlMap<**  
**Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR**  
**>** (p. 3716), **decaf::util::StlMap< int, Pointer< Command > >** (p. 3716), **decaf::util::StlMap<**  
**Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR**  
**>** (p. 3716), **decaf::util::StlMap< std::string, CachedProducer \* >** (p. 3716), and  
**decaf::util::StlMap< std::string, cms::Topic \* >** (p. 3716).

**6.501.3.9** **template<typename K, typename V, typename COMPARATOR = std::less<K>>**  
**virtual std::vector<K> decaf::util::Map< K, V, COMPARATOR >::keySet ( )**  
**const [pure virtual]**

Returns a **Set** (p. 3538) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 2223), **Set.remove** (p. 168), **removeAll**, **retainAll** and **clear** operations. It does not support the **add** or **addAll** operations.

## Returns

the entire set of keys in this map as a **std::vector**.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARA-**  
**TOR >** (p. 1273), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3716), **decaf::util::concurrent::Conc**  
**Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1273),  
**decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer<**  
**ConnectionState >, ConnectionId::COMPARATOR >** (p. 1273), **decaf::util::concurrent::ConcurrentStlM**  
**Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR**  
**>** (p. 1273), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >,**  
**Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1273), **decaf::util::concurrent::ConcurrentStl**  
**Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR**  
**>** (p. 1273), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId**  
**>, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1273), **decaf::util::StlMap<**  
**cms::Session \*, SessionResolver \* >** (p. 3716), **decaf::util::StlMap< std::string,**  
**WireFormatFactory \* >** (p. 3716), **decaf::util::StlMap< std::string, Primitive-**  
**ValueNode >** (p. 3716), **decaf::util::StlMap< std::string, cms::Queue \* >** (p. 3716),  
**decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*,**

## 6.501 decaf::util::Map< K, V, COMPARATOR > Class Template Reference 2549

**commands::ProducerId::COMPARATOR** > (p. 3716), **decaf::util::StlMap< std::string, CachedConsumer \* >** (p. 3716), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR >** (p. 3716), **decaf::util::StlMap< std::string, TransportFactory \* >** (p. 3716), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 3716), **decaf::util::StlMap< int, Pointer< Command > >** (p. 3716), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR >** (p. 3716), **decaf::util::StlMap< std::string, CachedProducer \* >** (p. 3716), and **decaf::util::StlMap< std::string, cms::Topic \* >** (p. 3716).

Referenced by **decaf::util::StlMap< std::string, cms::Topic \* >::equals()**, and **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals()**.

```
6.501.3.10  template<typename K, typename V, typename COMPARATOR
            = std::less<K>> virtual void decaf::util::Map< K, V,
            COMPARATOR >::put ( const K & key, const V & value ) throw (
            decaf::lang::exceptions::UnsupportedOperationException ) [pure
            virtual]
```

Sets the value for the specified key.

### Parameters

<i>key</i>	The target key.
<i>value</i>	The value to be set.

### Exceptions

<i>UnsupportedOperationException</i>	if this map is unmodifiable.
--------------------------------------	------------------------------

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1275), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3718), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1275), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1275), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1275), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1275), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 1275), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1275), **decaf::util::StlMap< cms::Session \*, SessionResolver \* >** (p. 3718), **decaf::util::StlMap< std::string, WireFormatFactory \* >** (p. 3718), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 3718), **decaf::util::StlMap< std::string, cms::Queue \* >** (p. 3718), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR >** (p. 3718), **decaf::util::StlMap< std::string, CachedConsumer \* >** (p. 3718), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR >** (p. 3718), **decaf::util::StlMap< std::string, TransportFactory \* >** (p. 3718), **decaf::util::StlMap<**

**Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR**  
 > (p. 3718), **decaf::util::StdMap< int, Pointer< Command > >** (p. 3718), **decaf::util::StdMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR**  
 > (p. 3718), **decaf::util::StdMap< std::string, CachedProducer \* >** (p. 3718), and  
**decaf::util::StdMap< std::string, cms::Topic \* >** (p. 3718).

6.501.3.11 **template<typename K, typename V, typename COMPARATOR =**  
**std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR**  
**>::putAll ( const Map< K, V, COMPARATOR > & other ) throw (**  
**decaf::lang::exceptions::UnsupportedOperationException )** [pure  
 virtual]

Stores a copy of the Mappings contained in the other **Map** (p. 2538) in this one.

#### Parameters

<i>other</i>	A <b>Map</b> (p. 2538) instance whose elements are to all be inserted in this <b>Map</b> (p. 2538).
--------------	---

#### Exceptions

<i>UnsupportedOperationException</i>	If the implementing class does not support the putAll operation.
--------------------------------------	--

Implemented in **decaf::util::concurrent::ConcurrentStdMap< K, V, COMPARA-**  
**TOR >** (p. 1276), **decaf::util::StdMap< K, V, COMPARATOR >** (p. 3718), **decaf::util::concurrent::Concu**  
**Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1276),  
**decaf::util::concurrent::ConcurrentStdMap< Pointer< ConnectionId >, Pointer<**  
**ConnectionState >, ConnectionId::COMPARATOR >** (p. 1276), **decaf::util::concurrent::ConcurrentStdM**  
**Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR**  
**>** (p. 1276), **decaf::util::concurrent::ConcurrentStdMap< Pointer< SessionId >,**  
**Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1276), **decaf::util::concurrent::ConcurrentStd**  
**Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR**  
**>** (p. 1276), **decaf::util::concurrent::ConcurrentStdMap< Pointer< ProducerId**  
**>, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1276), **decaf::util::StdMap<**  
**cms::Session \*, SessionResolver \* >** (p. 3718), **decaf::util::StdMap< std::string,**  
**WireFormatFactory \* >** (p. 3718), **decaf::util::StdMap< std::string, Primitive-**  
**ValueNode >** (p. 3718), **decaf::util::StdMap< std::string, cms::Queue \* >** (p. 3718),  
**decaf::util::StdMap< Pointer< commands::ProducerId >, ActiveMQProducer \*,**  
**commands::ProducerId::COMPARATOR >** (p. 3718), **decaf::util::StdMap< std::string,**  
**CachedConsumer \* >** (p. 3718), **decaf::util::StdMap< Pointer< commands::ConsumerId**  
**>, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR >** (p. 3718),  
**decaf::util::StdMap< std::string, TransportFactory \* >** (p. 3718), **decaf::util::StdMap<**  
**Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR**  
**>** (p. 3718), **decaf::util::StdMap< int, Pointer< Command > >** (p. 3718), **decaf::util::StdMap<**  
**Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR**  
**>** (p. 3718), **decaf::util::StdMap< std::string, CachedProducer \* >** (p. 3718), and  
**decaf::util::StdMap< std::string, cms::Topic \* >** (p. 3718).

## 6.501 decaf::util::Map< K, V, COMPARATOR > Class Template Reference 2551

6.501.3.12 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual V decaf::util::Map< K, V, COMPARATOR >::remove ( const K &  
key ) throw ( decaf::lang::exceptions::NoSuchElementException,  
decaf::lang::exceptions::UnsupportedOperationException ) [pure  
virtual]`

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

### Parameters

<i>key</i>	The search key.
------------	-----------------

### Returns

a copy of the element that was previously mapped to the given key

### Exceptions

<i>NoSuchElementException</i>	if this key is not in the <b>Map</b> (p. 2538).
<i>UnsupportedOperationException</i>	if this map is unmodifiable.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1278), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3719), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1278), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1278), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1278), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1278), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1278), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1278), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3719), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3719), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3719), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3719), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3719), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3719), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3719), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3719), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3719), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3719), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3719), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3719), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3719).

```
6.501.3.13  template<typename K, typename V, typename COMPARATOR = std::less<K>>
             virtual std::size_t decaf::util::Map< K, V, COMPARATOR >::size ( ) const
             [pure virtual]
```

### Returns

The number of elements (key/value pairs) in this map.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1279), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3719), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1279), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1279), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1279), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1279), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1279), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1279), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3719), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3719), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3719), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3719), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3719), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3719), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3719), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3719), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3719), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3719), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3719), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3719), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3719).

```
6.501.3.14  template<typename K, typename V, typename COMPARATOR = std::less<K>>
             virtual std::vector<V> decaf::util::Map< K, V, COMPARATOR >::values ( )
             const [pure virtual]
```

### Returns

the entire set of values in this map as a `std::vector`.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1280), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3720), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1280), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1280), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1280), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1280), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1280), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1280), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ActiveMQProducer >, Pointer< ActiveMQProducer >, ActiveMQProducer::COMPARATOR >` (p. 1280), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ActiveMQConsumer >, Pointer< ActiveMQConsumer >, ActiveMQConsumer::COMPARATOR >` (p. 1280), `decaf::util::concurrent::ConcurrentStlMap< Pointer< TransportFactory >, Pointer< TransportFactory >, TransportFactory::COMPARATOR >` (p. 1280), `decaf::util::concurrent::ConcurrentStlMap< Pointer< CachedProducer >, Pointer< CachedProducer >, CachedProducer::COMPARATOR >` (p. 1280), and `decaf::util::concurrent::ConcurrentStlMap< Pointer< cms::Topic >, Pointer< cms::Topic >, cms::Topic::COMPARATOR >` (p. 1280).



> (p. 1280), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ProducerId** >, **Pointer**< **ProducerState** >, **ProducerId::COMPARATOR** > (p. 1280), **decaf::util::StlMap**< **cms::Session** \*, **SessionResolver** \* > (p. 3720), **decaf::util::StlMap**< **std::string**, **WireFormatFactory** \* > (p. 3720), **decaf::util::StlMap**< **std::string**, **Primitive-ValueNode** > (p. 3720), **decaf::util::StlMap**< **std::string**, **cms::Queue** \* > (p. 3720), **decaf::util::StlMap**< **Pointer**< **commands::ProducerId** >, **ActiveMQProducer** \*, **commands::ProducerId::COMPARATOR** > (p. 3720), **decaf::util::StlMap**< **std::string**, **CachedConsumer** \* > (p. 3720), **decaf::util::StlMap**< **Pointer**< **commands::ConsumerId** >, **ActiveMQConsumer** \*, **commands::ConsumerId::COMPARATOR** > (p. 3720), **decaf::util::StlMap**< **std::string**, **TransportFactory** \* > (p. 3720), **decaf::util::StlMap**< **Pointer**< **ConsumerId** >, **Pointer**< **ConsumerInfo** >, **ConsumerId::COMPARATOR** > (p. 3720), **decaf::util::StlMap**< **int**, **Pointer**< **Command** > > (p. 3720), **decaf::util::StlMap**< **Pointer**< **commands::ConsumerId** >, **Dispatcher** \*, **commands::ConsumerId::COMPARATOR** > (p. 3720), **decaf::util::StlMap**< **std::string**, **CachedProducer** \* > (p. 3720), and **decaf::util::StlMap**< **std::string**, **cms::Topic** \* > (p. 3720).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Map.h`

## 6.502 cms::MapMessage Class Reference

A **MapMessage** (p. 2551) object is used to send a set of name-value pairs.

```
#include <src/main/cms/MapMessage.h>
```

Inheritance diagram for cms::MapMessage:

### Public Member Functions

- virtual **~MapMessage** ()
- virtual **std::vector**< **std::string** > **getMapNames** () const =0 throw ( **CMSException** )  
*Returns an Enumeration of all the names in the **MapMessage** (p. 2551) object.*
- virtual bool **itemExists** (const **std::string** &name) const =0 throw ( **CMSException** )  
*Indicates whether an item exists in this **MapMessage** (p. 2551) object.*
- virtual bool **getBoolean** (const **std::string** &name) const =0 throw ( **cms::MessageFormatException**, **cms::CMSException** )  
*Returns the Boolean value of the Specified name.*
- virtual void **setBoolean** (const **std::string** &name, bool value)=0 throw ( **cms::MessageNotWriteableException**, **cms::CMSException** )  
*Sets a boolean value with the specified name into the Map.*

- virtual unsigned char **getByte** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Byte value of the Specified name.*
- virtual void **setByte** (const std::string &name, unsigned char value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Byte value with the specified name into the Map.*
- virtual std::vector< unsigned char > **getBytes** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Bytes value of the Specified name.*
- virtual void **setBytes** (const std::string &name, const std::vector< unsigned char > &value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Bytes value with the specified name into the Map.*
- virtual char **getChar** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Char value of the Specified name.*
- virtual void **setChar** (const std::string &name, char value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Char value with the specified name into the Map.*
- virtual double **getDouble** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Double value of the Specified name.*
- virtual void **setDouble** (const std::string &name, double value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Double value with the specified name into the Map.*
- virtual float **getFloat** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Float value of the Specified name.*
- virtual void **setFloat** (const std::string &name, float value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Float value with the specified name into the Map.*
- virtual int **getInt** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Int value of the Specified name.*

- virtual void **setInt** (const std::string &name, int value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Int value with the specified name into the Map.*
- virtual long long **getLong** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Long value of the Specified name.*
- virtual void **setLong** (const std::string &name, long long value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Long value with the specified name into the Map.*
- virtual short **getShort** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Short value of the Specified name.*
- virtual void **setShort** (const std::string &name, short value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Short value with the specified name into the Map.*
- virtual std::string **getString** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the String value of the Specified name.*
- virtual void **setString** (const std::string &name, const std::string &value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a String value with the specified name into the Map.*

### 6.502.1 Detailed Description

A **MapMessage** (p. 2551) object is used to send a set of name-value pairs. The names are String objects, and the values are primitive data types in the Java programming language. The names must have a value that is not null, and not an empty string. The entries can be accessed sequentially or randomly by name. The order of the entries is undefined. **MapMessage** (p. 2551) inherits from the **Message** (p. 2614) interface and adds a message body that contains a Map.

When a client receives a **MapMessage** (p. 2551), it is in read-only mode. If a client attempts to write to the message at this point, a **MessageNotWriteableException** (p. 2808) is thrown. To place the **MapMessage** (p. 2551) back into a state where it can be read from and written to, call the `clearBody` method.

**MapMessage** (p. 2551) objects support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSEException** (p. 1190). The String-to-primitive conversions may throw a **MessageFormatException** (p. 2749) if the primitive's `valueOf()` method does not accept it as a valid String representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	char	int	long	float	double	String	byte[]
boolean	X								X	
byte		X	X		X	X			X	
short			X		X	X			X	
char				X					X	
int					X	X			X	
long						X			X	
float							X	X	X	
double								X	X	
String	X	X	X		X	X	X	X	X	
byte[]										X

Since

1.0

6.502.2 Constructor & Destructor Documentation

6.502.2.1 virtual cms::MapMessage::~MapMessage ( ) [inline, virtual]

6.502.3 Member Function Documentation

6.502.3.1 virtual bool cms::MapMessage::getBoolean ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSException ) [pure virtual]

Returns the Boolean value of the Specified name.

Parameters

name	Name of the value to fetch from the map
------	---

Exceptions

CMSException (p. 1190)	- if the operation fails due to an internal error.
MessageFormatException	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 355).

**6.502.3.2** `virtual unsigned char cms::MapMessage::getByte ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSException ) [pure virtual]`

Returns the Byte value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the operation fails due to an internal error.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 355).

**6.502.3.3** `virtual std::vector<unsigned char> cms::MapMessage::getBytes ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSException ) [pure virtual]`

Returns the Bytes value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the operation fails due to an internal error.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 356).

**6.502.3.4** `virtual char cms::MapMessage::getChar ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSException ) [pure virtual]`

Returns the Char value of the Specified name.

#### Parameters

<i>name</i>	name of the value to fetch from the map
-------------	---

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if the operation fails due to an internal error.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 356).

```
6.502.3.5  virtual double cms::MapMessage::getDouble ( const std::string & name ) const
            throw ( cms::MessageFormatException, cms::CMSEException )  [pure
                                virtual]
```

Returns the Double value of the Specified name.

**Parameters**

<i>name</i>	Name of the value to fetch from the map
-------------	---

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if the operation fails due to an internal error.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 357).

```
6.502.3.6  virtual float cms::MapMessage::getFloat ( const std::string & name ) const throw (
            cms::MessageFormatException, cms::CMSEException )  [pure
                                virtual]
```

Returns the Float value of the Specified name.

**Parameters**

<i>name</i>	Name of the value to fetch from the map
-------------	---

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if the operation fails due to an internal error.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 357).

**6.502.3.7** `virtual int cms::MapMessage::getInt ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSException ) [pure virtual]`

Returns the Int value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the operation fails due to an internal error.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 357).

**6.502.3.8** `virtual long long cms::MapMessage::getLong ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSException ) [pure virtual]`

Returns the Long value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the operation fails due to an internal error.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 358).

**6.502.3.9** `virtual std::vector< std::string > cms::MapMessage::getMapNames ( ) const throw ( CMSException ) [pure virtual]`

Returns an Enumeration of all the names in the **MapMessage** (p. 2551) object.

#### Returns

STL Vector of String values, each of which is the name of an item in the **MapMessage** (p. 2551)

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	- if the operation fails due to an internal error.
---	--

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 358).

**6.502.3.10** `virtual short cms::MapMessage::getShort ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSException ) [pure virtual]`

Returns the Short value of the Specified name.

**Parameters**

<i>name</i>	Name of the value to fetch from the map
-------------	---

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	- if the operation fails due to an internal error.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 359).

**6.502.3.11** `virtual std::string cms::MapMessage::getString ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSException ) [pure virtual]`

Returns the String value of the Specified name.

**Parameters**

<i>name</i>	Name of the value to fetch from the map
-------------	---

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	- if the operation fails due to an internal error.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 359).



**6.502.3.12** `virtual bool cms::MapMessage::itemExists ( const std::string & name ) const throw ( CMSEException ) [pure virtual]`

Indicates whether an item exists in this **MapMessage** (p. 2551) object.

#### Parameters

<i>name</i>	String name of the Object in question
-------------	---------------------------------------

#### Returns

boolean value indicating if the name is in the map

#### Exceptions

<b>CMSEException</b> (p. 1190)	- if the operation fails due to an internal error.
-----------------------------------	--

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 360).

**6.502.3.13** `virtual void cms::MapMessage::setBoolean ( const std::string & name, bool value ) throw ( cms::MessageNotWriteableException, cms::CMSEException ) [pure virtual]`

Sets a boolean value with the specified name into the Map.

#### Parameters

<i>name</i>	the name of the boolean
<i>value</i>	the boolean value to set in the Map

#### Exceptions

<b>CMSEException</b> (p. 1190)	- if the operation fails due to an internal error.
<b>MessageNotWritableException</b>	- if the <b>Message</b> (p. 2614) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 360).

**6.502.3.14** `virtual void cms::MapMessage::setByte ( const std::string & name, unsigned char value ) throw ( cms::MessageNotWriteableException, cms::CMSEException ) [pure virtual]`

Sets a Byte value with the specified name into the Map.

#### Parameters

<i>name</i>	the name of the Byte
<i>value</i>	the Byte value to set in the Map

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if the operation fails due to an internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the <b>Message</b> (p. 2614) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 360).

```
6.502.3.15 virtual void cms::MapMessage::setBytes ( const std::string &
name, const std::vector< unsigned char > & value ) throw (
cms::MessageNotWriteableException, cms::CMSEException ) [pure
virtual]
```

Sets a Bytes value with the specified name into the Map.

**Parameters**

<i>name</i>	The name of the Bytes
<i>value</i>	The Bytes value to set in the Map

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if the operation fails due to an internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the <b>Message</b> (p. 2614) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 361).

```
6.502.3.16 virtual void cms::MapMessage::setChar ( const std::string & name, char value )
throw ( cms::MessageNotWriteableException, cms::CMSEException )
[pure virtual]
```

Sets a Char value with the specified name into the Map.

**Parameters**

<i>name</i>	the name of the Char
<i>value</i>	the Char value to set in the Map

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if the operation fails due to an internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the <b>Message</b> (p. 2614) is in Read-only Mode.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 361).

**6.502.3.17** `virtual void cms::MapMessage::setDouble ( const std::string & name, double value )  
throw ( cms::MessageNotWriteableException, cms::CMSException )  
[pure virtual]`

Sets a Double value with the specified name into the Map.

#### Parameters

<i>name</i>	The name of the Double
<i>value</i>	The Double value to set in the Map

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the operation fails due to an internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the <b>Message</b> (p. 2614) is in Read-only Mode.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 362).

**6.502.3.18** `virtual void cms::MapMessage::setFloat ( const std::string & name, float value )  
throw ( cms::MessageNotWriteableException, cms::CMSException )  
[pure virtual]`

Sets a Float value with the specified name into the Map.

#### Parameters

<i>name</i>	The name of the Float
<i>value</i>	The Float value to set in the Map

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the operation fails due to an internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the <b>Message</b> (p. 2614) is in Read-only Mode.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 362).

**6.502.3.19** `virtual void cms::MapMessage::setInt ( const std::string & name, int value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Sets a Int value with the specified name into the Map.

**Parameters**

<i>name</i>	The name of the Int
<i>value</i>	The Int value to set in the Map

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if the operation fails due to an internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the <b>Message</b> (p. 2614) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 362).

```
6.502.3.20 virtual void cms::MapMessage::setLong ( const std::string & name, long long value
) throw ( cms::MessageNotWriteableException, cms::CMSEException )
[pure virtual]
```

Sets a Long value with the specified name into the Map.

**Parameters**

<i>name</i>	The name of the Long
<i>value</i>	The Long value to set in the Map

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if the operation fails due to an internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the <b>Message</b> (p. 2614) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 363).

```
6.502.3.21 virtual void cms::MapMessage::setShort ( const std::string & name, short value )
throw ( cms::MessageNotWriteableException, cms::CMSEException )
[pure virtual]
```

Sets a Short value with the specified name into the Map.

**Parameters**

<i>name</i>	The name of the Short
<i>value</i>	The Short value to set in the Map

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if the operation fails due to an internal error.
--	--

<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the <b>Message</b> (p. 2614) is in Read-only Mode.
---	---

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 363).

**6.502.3.22** virtual void cms::MapMessage::setString ( const std::string & name, const std::string & value ) throw ( cms::MessageNotWriteableException, cms::CMSEException ) [pure virtual]

Sets a String value with the specified name into the Map.

#### Parameters

<i>name</i>	The name of the String
<i>value</i>	The String value to set in the Map

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if the operation fails due to an internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the <b>Message</b> (p. 2614) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 364).

The documentation for this class was generated from the following file:

- src/main/cms/MapMessage.h

## 6.503 decaf::util::logging::MarkBlockLogger Class Reference

Defines a class that can be used to mark the entry and exit from scoped blocks.

```
#include <src/main/decaf/util/logging/MarkBlockLogger.h>
```

#### Public Member Functions

- **MarkBlockLogger** (**Logger** \*logger, const std::string &blockName)  
*Constructor - Marks Block entry.*
- virtual ~**MarkBlockLogger** ()

### 6.503.1 Detailed Description

Defines a class that can be used to mark the entry and exit from scoped blocks. Create an instance of this class at the start of a scoped block, passing it the logger to use and the name of the block. The block entry and exit will be marked using the scope name, logger to the logger at the MARKBLOCK log level.

### 6.503.2 Constructor & Destructor Documentation

**6.503.2.1** `decaf::util::logging::MarkBlockLogger::MarkBlockLogger ( Logger * logger, const std::string & blockName )` [inline]

Constructor - Marks Block entry.

#### Parameters

<i>logger</i>	<b>Logger</b> (p. 2461) to use
<i>blockName</i>	Block name

**6.503.2.2** `virtual decaf::util::logging::MarkBlockLogger::~MarkBlockLogger ( )` [inline, virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/MarkBlockLogger.h`

## 6.504 activemq::wireformat::MarshalAware Class Reference

```
#include <src/main/activemq/wireformat/MarshalAware.h>
```

Inheritance diagram for `activemq::wireformat::MarshalAware`:

### Public Member Functions

- virtual `~MarshalAware ()`
- virtual `bool isMarshalAware () const =0`  
*Determine if the class implementing this interface is really wanting to be told about marshaling.*
- virtual `void beforeMarshal (WireFormat *wireFormat)=0 throw ( decaf::io::IOException )`  
*Called before marshaling is started to prepare the object to be marshaled.*

- virtual void **afterMarshal** (**WireFormat** \*wireFormat)=0 throw ( decaf::io::IOException )  
*Called after marshaling is started to cleanup the object being marshaled.*
- virtual void **beforeUnmarshal** (**WireFormat** \*wireFormat)=0 throw ( decaf::io::IOException )  
*Called before unmarshaling is started to prepare the object to be unmarshaled.*
- virtual void **afterUnmarshal** (**WireFormat** \*wireFormat)=0 throw ( decaf::io::IOException )  
*Called after unmarshaling is started to cleanup the object being unmarshaled.*
- virtual void **setMarshaledForm** (**WireFormat** \*wireFormat, const std::vector< char > &data)=0  
*Called to set the data to this object that will contain the objects marshaled form.*
- virtual std::vector< unsigned char > **getMarshaledForm** (**WireFormat** \*wireFormat)=0  
*Called to get the data to this object that will contain the objects marshaled form.*

## 6.504.1 Constructor & Destructor Documentation

6.504.1.1 virtual activemq::wireformat::MarshalAware::~~MarshalAware ( ) [inline, virtual]

## 6.504.2 Member Function Documentation

6.504.2.1 virtual void activemq::wireformat::MarshalAware::afterMarshal ( **WireFormat** \* wireFormat ) throw ( decaf::io::IOException ) [pure virtual]

Called after marshaling is started to cleanup the object being marshaled.

### Parameters

<i>wireFormat</i>	- the wireformat object to control marshaling
-------------------	---

6.504.2.2 virtual void activemq::wireformat::MarshalAware::afterUnmarshal ( **WireFormat** \* wireFormat ) throw ( decaf::io::IOException ) [pure virtual]

Called after unmarshaling is started to cleanup the object being unmarshaled.

### Parameters

<i>wireFormat</i>	- the wireformat object to control unmarshaling
-------------------	---

**6.504.2.3** `virtual void activemq::wireformat::MarshalAware::beforeMarshal ( WireFormat * wireFormat ) throw ( decaf::io::IOException ) [pure virtual]`

Called before marshaling is started to prepare the object to be marshaled.

#### Parameters

<i>wireFormat</i>	- the wireformat object to control marshaling
-------------------	---

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 353), and **activemq::commands::ActiveMQTextMessage** (p. 668).

**6.504.2.4** `virtual void activemq::wireformat::MarshalAware::beforeUnmarshal ( WireFormat * wireFormat ) throw ( decaf::io::IOException ) [pure virtual]`

Called before unmarshaling is started to prepare the object to be unmarshaled.

#### Parameters

<i>wireFormat</i>	- the wireformat object to control unmarshaling
-------------------	---

**6.504.2.5** `virtual std::vector<unsigned char> activemq::wireformat::MarshalAware::getMarshaledForm ( WireFormat * wireFormat ) [pure virtual]`

Called to get the data to this object that will contain the objects marshaled form.

#### Parameters

<i>wireFormat</i>	- the wireformat object to control unmarshaling
-------------------	---

#### Returns

buffer that holds the objects data.

**6.504.2.6** `virtual bool activemq::wireformat::MarshalAware::isMarshalAware ( ) const [pure virtual]`

Determine if the class implementing this interface is really wanting to be told about marshaling.

Normally if you didn't want to be marshal aware you just wouldn't implement this interface but since this is C++ and we don't have true interfaces we need a flat inheritance hierarchy, so we always implement this.

#### Returns

true if this class cares about marshaling.



## 6.505 activemq::wireformat::openwire::marshal::v6::MarshallerFactory Class Reference 2569

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 359), `activemq::commands::BaseDataStructure` (p. 840), `activemq::commands::Message` (p. 2607), and `activemq::commands::WireFormatInfo` (p. 4100).

**6.504.2.7** `virtual void activemq::wireformat::MarshalAware::setMarshaledForm ( WireFormat * wireFormat, const std::vector< char > & data ) [pure virtual]`

Called to set the data to this object that will contain the objects marshaled form.

### Parameters

<i>wireFormat</i>	- the wireformat object to control unmarshaling
<i>data</i>	- vector of object binary data

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/MarshalAware.h`

## 6.505 activemq::wireformat::openwire::marshal::v6::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/MarshallerFactory.h>
```

### Public Member Functions

- `virtual ~MarshallerFactory ()`
- `virtual void configure (OpenWireFormat *format)`

### 6.505.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

### 6.505.2 Constructor & Destructor Documentation

6.505.2.1 `virtual activemq::wireformat::openwire::marshal::v6::MarshallerFactory::~MarshallerFactory ( ) [inline, virtual]`

### 6.505.3 Member Function Documentation

6.505.3.1 `virtual void activemq::wireformat::openwire::marshal::v6::MarshallerFactory::configure ( OpenWireFormat * format ) [virtual]`

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/MarshallerFactory.h`

## 6.506 activemq::wireformat::openwire::marshal::v3::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h>
```

### Public Member Functions

- `virtual ~MarshallerFactory ()`
- `virtual void configure (OpenWireFormat *format)`

### 6.506.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

### 6.506.2 Constructor & Destructor Documentation

6.506.2.1 `virtual activemq::wireformat::openwire::marshal::v3::MarshallerFactory::~MarshallerFactory ( ) [inline, virtual]`

### 6.506.3 Member Function Documentation

6.506.3.1 `virtual void activemq::wireformat::openwire::marshal::v3::MarshallerFactory::configure ( OpenWireFormat * format ) [virtual]`

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h`

## 6.507 activemq::wireformat::openwire::marshal::v4::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h>
```

### Public Member Functions

- virtual `~MarshallerFactory()`
- virtual void `configure(OpenWireFormat *format)`

#### 6.507.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

#### 6.507.2 Constructor & Destructor Documentation

6.507.2.1 virtual `activemq::wireformat::openwire::marshal::v4::MarshallerFactory::~MarshallerFactory()` [`inline`, `virtual`]

#### 6.507.3 Member Function Documentation

6.507.3.1 virtual void `activemq::wireformat::openwire::marshal::v4::MarshallerFactory::configure(OpenWireFormat *format)` [`virtual`]

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h`

## 6.508 activemq::wireformat::openwire::marshal::v5::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MarshallerFactory.h>
```

### Public Member Functions

- virtual `~MarshallerFactory()`
- virtual void `configure(OpenWireFormat *format)`

### 6.508.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

### 6.508.2 Constructor & Destructor Documentation

6.508.2.1 `virtual activemq::wireformat::openwire::marshal::v5::MarshallerFactory::~MarshallerFactory ( ) [inline, virtual]`

### 6.508.3 Member Function Documentation

6.508.3.1 `virtual void activemq::wireformat::openwire::marshal::v5::MarshallerFactory::configure ( OpenWireFormat * format ) [virtual]`

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/MarshallerFactory.h`

## 6.509 activemq::wireformat::openwire::marshal::v1::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MarshallerFacto
```

### Public Member Functions

- `virtual ~MarshallerFactory ()`
- `virtual void configure (OpenWireFormat *format)`

### 6.509.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

## 6.509.2 Constructor & Destructor Documentation

6.509.2.1 virtual activemq::wireformat::openwire::marshal::v1::MarshallerFactory::~MarshallerFactory ( ) [inline, virtual]

## 6.509.3 Member Function Documentation

6.509.3.1 virtual void activemq::wireformat::openwire::marshal::v1::MarshallerFactory::configure ( OpenWireFormat \* *format* ) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/MarshallerFactory.h

## 6.510 activemq::wireformat::openwire::marshal::v2::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MarshallerFactory.h>
```

### Public Member Functions

- virtual ~MarshallerFactory ()
- virtual void **configure** (OpenWireFormat \*format)

### 6.510.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

## 6.510.2 Constructor & Destructor Documentation

6.510.2.1 virtual activemq::wireformat::openwire::marshal::v2::MarshallerFactory::~MarshallerFactory ( ) [inline, virtual]

## 6.510.3 Member Function Documentation

6.510.3.1 virtual void activemq::wireformat::openwire::marshal::v2::MarshallerFactory::configure ( OpenWireFormat \* *format* ) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/MarshallerFactory.h

## 6.511 activemq::util::MarshallingSupport Class Reference

```
#include <src/main/activemq/util/MarshallingSupport.h>
```

### Public Member Functions

- **MarshallingSupport** ()
- virtual **~MarshallingSupport** ()

### Static Public Member Functions

- static void **writeString** (decaf::io::DataOutputStream &dataOut, const std::string &value) throw ( decaf::io::IOException )  
*Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.*
- static void **writeString16** (decaf::io::DataOutputStream &dataOut, const std::string &value) throw ( decaf::io::IOException )  
*Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.*
- static void **writeString32** (decaf::io::DataOutputStream &dataOut, const std::string &value) throw ( decaf::io::IOException )  
*Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.*
- static std::string **readString16** (decaf::io::DataInputStream &dataIn) throw ( decaf::io::IOException )  
*Reads an Openwire encoded string from the provided DataInputStream.*
- static std::string **readString32** (decaf::io::DataInputStream &dataIn) throw ( decaf::io::IOException )  
*Reads an Openwire encoded string from the provided DataInputStream.*
- static std::string **asciiToModifiedUtf8** (const std::string &asciiString) throw ( decaf::io::UTFDataFormatException )  
*Given an ASCII String with byte values [0..255] convert the string to a string containing the modified UTF-8 form of that same string.*
- static std::string **modifiedUtf8ToAscii** (const std::string modifiedUtf8String) throw ( decaf::io::UTFDataFormatException )  
*Given a string that contains bytes in the Java Modified UTF-8 format convert that string back into ASCII values from [0..255].*

### 6.511.1 Constructor & Destructor Documentation

6.511.1.1 `activemq::util::MarshallingSupport::MarshallingSupport ( )`

6.511.1.2 `virtual activemq::util::MarshallingSupport::~~MarshallingSupport ( )` `[virtual]`

### 6.511.2 Member Function Documentation

6.511.2.1 `static std::string activemq::util::MarshallingSupport::asciiToModifiedUtf8 ( const std::string & asciiString ) throw ( decaf::io::UTFDataFormatException )`  
`[static]`

Given an ASCII String with byte values [0..255] convert the string to a string containing the modified UTF-8 form of that same string.

This allows an ASCII string containing values greater than 127 as well as embedded NULLs to be sent to a Java client.

#### Parameters

<i>asciiString</i>	The ASCII string to encode as Modified UTF-8
--------------------	--

#### Returns

a string containing the Modified UTF-8 encoded form of the provided string.

#### Exceptions

<i>UTFDataFormatException</i>	if the length of the encoded string would exceed the size of an signed integer.
-------------------------------	---

6.511.2.2 `static std::string activemq::util::MarshallingSupport::modifiedUtf8ToAscii ( const std::string modifiedUtf8String ) throw ( decaf::io::UTFDataFormatException )`  
`[static]`

Given a string that contains bytes in the Java Modified UTF-8 format convert that string back into ASCII values from [0..255].

This will handle any string sent from a Java client which contains values within the [0..255] range or has embedded Nulls. Strings that have encoded values greater than 255 will cause an exception to be thrown.

#### Parameters

<i>modifiedUtf8String</i>	The string to convert from Modified UTF-8 to ASCII.
---------------------------	---

#### Returns

the ASCII encoded version of the provided string.

**Exceptions**

<i>UTFDataFormatException</i>	if the provided string contains invalid data or the character values encoded in the string exceed ASCII value 255.
-------------------------------	--

**6.511.2.3** `static std::string activemq::util::MarshallingSupport::readString16 (`  
`decaf::io::DataInputStream & dataIn ) throw ( decaf::io::IOException )`  
`[static]`

Reads an Openwire encoded string from the provided DataInputStream.

No string processing is performed by this method, clients that know the data contains UTF-8 encoded content must use one of the utility methods of this class to decode the UTF-8 data.

This version assumes a size prefix of 16bits.

**Parameters**

<i>dataIn</i>	The DataInputStream to read the String data from.
---------------	---

**Returns**

the String value.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while writing the string.
--------------------	--

**6.511.2.4** `static std::string activemq::util::MarshallingSupport::readString32 (`  
`decaf::io::DataInputStream & dataIn ) throw ( decaf::io::IOException )`  
`[static]`

Reads an Openwire encoded string from the provided DataInputStream.

No string processing is performed by this method, clients that know the data contains UTF-8 encoded content must use one of the utility methods of this class to decode the UTF-8 data.

This version assumes a size prefix of 32bits.

**Parameters**

<i>dataIn</i>	The DataInputStream to read the String data from.
---------------	---

**Returns**

the String value.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while writing the string.
--------------------	--



**6.511.2.5** static void activemq::util::MarshallingSupport::writeString (  
decaf::io::DataOutputStream & *dataOut*, const std::string & *value* ) throw (  
decaf::io::IOException) [static]

Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.

User must encode to Modified UTF-8 as needed.

#### Parameters

<i>dataOut</i>	The DataOutputStream to write the String data to.
<i>value</i>	Thre String value to write in Openwire form.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while writing the string.
--------------------	--

**6.511.2.6** static void activemq::util::MarshallingSupport::writeString16 (  
decaf::io::DataOutputStream & *dataOut*, const std::string & *value* ) throw (  
decaf::io::IOException) [static]

Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.

User must encode to Modified UTF-8 as needed. This method write out only the size as a short and the string data no Openwire Type tag is appended.

#### Parameters

<i>dataOut</i>	The DataOutputStream to write the String data to.
<i>value</i>	Thre String value to write in Openwire form.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while writing the string.
--------------------	--

**6.511.2.7** static void activemq::util::MarshallingSupport::writeString32 (  
decaf::io::DataOutputStream & *dataOut*, const std::string & *value* ) throw (  
decaf::io::IOException) [static]

Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.

User must encode to Modified UTF-8 as needed. This method write out only the size as a int and the string data no Openwire Type tag is appended.

#### Parameters

<i>dataOut</i>	The DataOutputStream to write the String data to.
<i>value</i>	Thre String value to write in Openwire form.

### Exceptions

<i>IOException</i> if an I/O error occurs while writing the string.
---

The documentation for this class was generated from the following file:

- `src/main/activemq/util/MarshallingSupport.h`

## 6.512 decaf::lang::Math Class Reference

The class **Math** (p. 2575) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

```
#include <src/main/decaf/lang/Math.h>
```

### Public Member Functions

- **Math** ()
- virtual  $\sim$ **Math** ()

### Static Public Member Functions

- static int **abs** (int value)  
*Returns the absolute value of an int value.*
- static long long **abs** (long long value)  
*Returns the absolute value of an long long value.*
- static float **abs** (float value)  
*Returns the absolute value of a float value.*
- static double **abs** (double value)  
*Returns the absolute value of a double value.*
- static double **sqrt** (double value)  
*Returns the arc cosine of an angle, in the range of 0.0 through pi.*
- static double **pow** (double base, double exp)  
*Returns the value of the first argument raised to the power of the second argument.*
- static short **min** (short a, short b)  
*Returns the double value that is closest in value to the argument and is equal to a mathematical integer.*
- static int **min** (int a, int b)

*Returns the smaller of two `int` values.*

- static unsigned int **min** (unsigned int a, unsigned int b)

*Returns the smaller of two unsigned `int` values.*

- static long long **min** (long long a, long long b)

*Returns the smaller of two long `long` values.*

- static float **min** (float a, float b)

*Returns the smaller of two float values.*

- static double **min** (double a, double b)

*Returns the smaller of two double values.*

- static short **max** (short a, short b)

*Returns the larger of two short values.*

- static int **max** (int a, int b)

*Returns the larger of two `int` values.*

- static long long **max** (long long a, long long b)

*Returns the larger of two long `long` values.*

- static float **max** (float a, float b)

*Returns the greater of two float values.*

- static double **max** (double a, double b)

*Returns the greater of two double values.*

- static double **ceil** (double value)

*Returns the natural logarithm (base  $e$ ) of a double value.*

- static double **floor** (double value)

*Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer.*

- static int **round** (float value)

*Returns the closest int to the argument.*

- static long long **round** (double value)

*Returns the closest long long to the argument.*

- static double **random** ()

*Computes the remainder operation on two arguments as prescribed by the IEEE 754 standard.*

- static float **signum** (float value)

*Returns Euler's number  $e$  raised to the power of a double value.*

- static double **signum** (double value)

*Returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero.*

- static double **toRadians** (double angdeg)

*Returns the measure in radians of the supplied degree angle.*

- static double **toDegrees** (double angrad)

*Returns the measure in degrees of the supplied radian angle.*

### Static Public Attributes

- static const double **E**
- static const double **PI**

### 6.512.1 Detailed Description

The class **Math** (p.2575) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

### 6.512.2 Constructor & Destructor Documentation

6.512.2.1 `decaf::lang::Math::Math ( )` [inline]

6.512.2.2 `virtual decaf::lang::Math::~~Math ( )` [inline, virtual]

### 6.512.3 Member Function Documentation

6.512.3.1 `static int decaf::lang::Math::abs ( int value )` [inline, static]

Returns the absolute value of an int value.

If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned.

#### Parameters

<i>value</i>	- the value to return the abs of
--------------	----------------------------------

#### Returns

the value if positive, otherwise the negative of value

**6.512.3.2** `static long long decaf::lang::Math::abs ( long long value )` `[inline, static]`

Returns the absolute value of an long long value.

If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned.

#### Parameters

<i>value</i>	- the value to return the abs of
--------------	----------------------------------

#### Returns

the value if positive, otherwise the negative of value

**6.512.3.3** `static double decaf::lang::Math::abs ( double value )` `[static]`

Returns the absolute value of a double value.

If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned. Special cases:

o If the argument is positive zero or negative zero, the result is positive zero. o If the argument is infinite, the result is positive infinity. o If the argument is NaN, the result is NaN.

In other words, the result is the same as the value of the expression: **Double::longBitsToDouble** (p. 1848)( 0x7fffffffffffffffULL & Double::doubleToLongBits( *value* ) )

#### Parameters

<i>value</i>	- the value to return the abs of
--------------	----------------------------------

#### Returns

the value if positive, otherwise the negative of value

**6.512.3.4** `static float decaf::lang::Math::abs ( float value )` `[static]`

Returns the absolute value of a float value.

If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned. Special cases:

o If the argument is positive zero or negative zero, the result is positive zero. o If the argument is infinite, the result is positive infinity. o If the argument is NaN, the result is NaN.

In other words, the result is the same as the value of the expression: **Float::intBitsToFloat** (p. 1967)( 0x7fffffff & Float::floatToIntBits( *value* ) )

**Parameters**

<i>value</i>	- the value to return the abs of
--------------	----------------------------------

**Returns**

the value if positive, otherwise the negative of value

**6.512.3.5 static double decaf::lang::Math::ceil ( double *value* ) [static]**

Returns the natural logarithm (base e) of a double value.

Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is negative infinity.

**Parameters**

<i>value</i>	the value to compute the natural log of.
--------------	--

**Returns**

the natural log of value. Returns the base 10 logarithm of a double value. Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is negative infinity. o If the argument is equal to  $10^n$  for integer n, then the result is n.

**Parameters**

<i>value</i>	- the value to operate on
--------------	---------------------------

**Returns**

the long base 10 of value Returns the natural logarithm of the sum of the argument and 1. Note that for small values x, the result of  $\log_{10}(x)$  is much closer to the true result of  $\ln(1 + x)$  than the floating-point evaluation of  $\log(1.0+x)$ .

Special cases:

o If the argument is NaN or less than -1, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative one, then the result is negative infinity. o If the argument is zero, then the result is a zero with the same sign as the argument.

**Parameters**

<i>value</i>	- the value to operate on
--------------	---------------------------

**Returns**

the the value  $\ln(x + 1)$ , the natural log of  $x + 1$  Returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer. Special cases:

- o If the argument value is already equal to a mathematical integer, then the result is the same as the argument.
- o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument.
- o If the argument value is less than zero but greater than -1.0, then the result is negative zero.

Note that the value of `Math.ceil(x)` is exactly the value of `-Math.floor(-x)`.

**Parameters**

<i>value</i>	- the value to find the ceiling of
--------------	------------------------------------

**Returns**

the smallest (closest to negative infinity) floating-point value that is greater than or equal to the argument and is equal to a mathematical integer.

**6.512.3.6 static double decaf::lang::Math::floor ( double *value* ) [static]**

Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer.

Special cases:

- o If the argument value is already equal to a mathematical integer, then the result is the same as the argument.
- o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument.

**Parameters**

<i>value</i>	- the value to find the floor of
--------------	----------------------------------

**Returns**

the largest (closest to positive infinity) floating-point value that less than or equal to the argument and is equal to a mathematical integer.

**6.512.3.7 static float decaf::lang::Math::max ( float *a*, float *b* ) [static]**

Returns the greater of two float values.

That is, the result is the argument closer to positive infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other negative zero, the result is positive zero.

**Parameters**

<i>a</i>	- an argument.
<i>b</i>	- another argument.

**Returns**

the larger of *a* and *b*.

**6.512.3.8 static double decaf::lang::Math::max ( double *a*, double *b* ) [static]**

Returns the greater of two double values.

That is, the result is the argument closer to positive infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other negative zero, the result is positive zero.

**Parameters**

<i>a</i>	- an argument.
<i>b</i>	- another argument.

**Returns**

the larger of *a* and *b*.

**6.512.3.9 static short decaf::lang::Math::max ( short *a*, short *b* ) [inline, static]**

Returns the larger of two short values.

That is, the result the argument closer to the value of **Short::MAX\_VALUE** (p.3548). If the arguments have the same value, the result is that same value.

**Parameters**

<i>a</i>	- an argument.
<i>b</i>	- another argument.

**Returns**

the larger of *a* and *b*.

**6.512.3.10 static int decaf::lang::Math::max ( int *a*, int *b* ) [inline, static]**

Returns the larger of two int values.

That is, the result the argument closer to the value of **Integer::MAX\_VALUE** (p.2159). If the arguments have the same value, the result is that same value.



**Parameters**

<i>a</i>	- an argument.
<i>b</i>	- another argument.

**Returns**

the larger of *a* and *b*.

**6.512.3.11** `static long long decaf::lang::Math::max ( long long a, long long b )` [`inline`, `static`]

Returns the larger of two `long long` values.

That is, the result the argument closer to the value of **Long : :MAX\_VALUE** (p. 2510).  
If the arguments have the same value, the result is that same value.

**Parameters**

<i>a</i>	- an argument.
<i>b</i>	- another argument.

**Returns**

the larger of *a* and *b*.

**6.512.3.12** `static int decaf::lang::Math::min ( int a, int b )` [`inline`, `static`]

Returns the smaller of two `int` values.

That is, the result the argument closer to the value of **Integer : :MIN\_VALUE** (p. 2159).  
If the arguments have the same value, the result is that same value.

**Parameters**

<i>a</i>	- an argument.
<i>b</i>	- another argument.

**Returns**

the smaller of *a* and *b*.

**6.512.3.13** `static unsigned int decaf::lang::Math::min ( unsigned int a, unsigned int b )`  
[`inline`, `static`]

Returns the smaller of two `unsigned int` values.

That is, the result the argument closer to the value of **Integer : :MIN\_VALUE** (p. 2159).  
If the arguments have the same value, the result is that same value.

**Parameters**

<i>a</i>	- an argument.
<i>b</i>	- another argument.

**Returns**

the smaller of *a* and *b*.

**6.512.3.14** `static long long decaf::lang::Math::min ( long long a, long long b )` `[inline, static]`

Returns the smaller of two `long long` values.

That is, the result the argument closer to the value of **Long: :MIN\_VALUE** (p.2510). If the arguments have the same value, the result is that same value.

**Parameters**

<i>a</i>	- an argument.
<i>b</i>	- another argument.

**Returns**

the smaller of *a* and *b*.

**6.512.3.15** `static float decaf::lang::Math::min ( float a, float b )` `[static]`

Returns the smaller of two `float` values.

That is, the result is the value closer to negative infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other is negative zero, the result is negative zero.

**Parameters**

<i>a</i>	- an argument.
<i>b</i>	- another argument.

**Returns**

the smaller of *a* and *b*.

**6.512.3.16** `static double decaf::lang::Math::min ( double a, double b )` `[static]`

Returns the smaller of two `double` values.

That is, the result is the value closer to negative infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN.

Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other is negative zero, the result is negative zero.

#### Parameters

<i>a</i>	- an argument.
<i>b</i>	- another argument.

#### Returns

the smaller of *a* and *b*.

#### 6.512.3.17 static short decaf::lang::Math::min ( short *a*, short *b* ) [inline, static]

Returns the double value that is closest in value to the argument and is equal to a mathematical integer.

If two double values that are mathematical integers are equally close, the result is the integer value that is even. Special cases:

o If the argument value is already equal to a mathematical integer, then the result is the same as the argument. o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument.

#### Parameters

<i>value</i>	- the value to round to the nearest integer
--------------	---

#### Returns

the rounded value Returns the smaller of two short values. That is, the result is the argument closer to the value of **decaf.lang.Short::MIN\_VALUE** (p. 3548) . If the arguments have the same value, the result is that same value.

#### Parameters

<i>a</i>	- an argument.
<i>b</i>	- another argument.

#### Returns

the smaller of *a* and *b*.

#### 6.512.3.18 static double decaf::lang::Math::pow ( double *base*, double *exp* ) [static]

Returns the value of the first argument raised to the power of the second argument.

Special cases:

o If the second argument is positive or negative zero, then the result is 1.0. o If the second argument is 1.0, then the result is the same as the first argument. o If the second

argument is NaN, then the result is NaN. o If the first argument is NaN and the second argument is nonzero, then the result is NaN.

#### Parameters

<i>base</i>	- the base
<i>exp</i>	- the exponent

#### Returns

the base raised to the power of exp.

#### 6.512.3.19 static double decaf::lang::Math::random ( ) [static]

Computes the remainder operation on two arguments as prescribed by the IEEE 754 standard.

The remainder value is mathematically equal to  $f1 - f2 \times n$ , where  $n$  is the mathematical integer closest to the exact mathematical value of the quotient  $f1/f2$ , and if two mathematical integers are equally close to  $f1/f2$ , then  $n$  is the integer that is even. If the remainder is zero, its sign is the same as the sign of the first argument. Special cases:

o If either argument is NaN, or the first argument is infinite, or the second argument is positive zero or negative zero, then the result is NaN. o If the first argument is finite and the second argument is infinite, then the result is the same as the first argument.

#### Parameters

<i>f1</i>	- the dividend.
<i>f2</i>	- the divisor

#### Returns

the IEEE remainder of value Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0. Returned values are chosen pseudorandomly with (approximately) uniform distribution from that range.

When this method is first called, it creates a single new pseudorandom-number generator; This new pseudorandom-number generator is used thereafter for all calls to this method and is used nowhere else.

This method is properly synchronized to allow correct use by more than one thread. However, if many threads need to generate pseudorandom numbers at a great rate, it may reduce contention for each thread to have its own pseudorandom-number generator.

#### Returns

a pseudorandom double greater than or equal to 0.0 and less than 1.0.

**6.512.3.20 static long long decaf::lang::Math::round ( double *value* ) [static]**

Returns the closest long long to the argument.

The result is rounded to an integer by adding 1/2, taking the floor of the result, and casting the result to type long long. In other words, the result is equal to the value of the expression: (long long)**Math.floor** (p. 2580)(a + 0.5d)

o If the argument is NaN, the result is 0. o If the argument is negative infinity or any value less than or equal to the value of **Long::MIN\_VALUE** (p. 2510), the result is equal to the value of **Long::MIN\_VALUE** (p. 2510). o If the argument is positive infinity or any value greater than or equal to the value of **Long::MAX\_VALUE** (p. 2510), the result is equal to the value of **Long::MAX\_VALUE** (p. 2510).

**Parameters**

<i>value</i>	- the value to round
--------------	----------------------

**Returns**

the value of the argument rounded to the nearest integral value.

**6.512.3.21 static int decaf::lang::Math::round ( float *value* ) [static]**

Returns the closest int to the argument.

The result is rounded to an integer by adding 1/2, taking the floor of the result, and casting the result to type int. In other words, the result is equal to the value of the expression: (int)**Math.floor** (p. 2580)( a + 0.5f )

o If the argument is NaN, the result is 0. o If the argument is negative infinity or any value less than or equal to the value of **Integer::MIN\_VALUE** (p. 2159), the result is equal to the value of **Integer::MIN\_VALUE** (p. 2159). o If the argument is positive infinity or any value greater than or equal to the value of **Integer::MAX\_VALUE** (p. 2159), the result is equal to the value of **Integer::MAX\_VALUE** (p. 2159).

**Parameters**

<i>value</i>	- the value to round
--------------	----------------------

**Returns**

the value of the argument rounded to the nearest integral value.

**6.512.3.22 static double decaf::lang::Math::signum ( double *value* ) [static]**

Returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero.

Special Cases:

o If the argument is NaN, then the result is NaN. o If the argument is positive zero or negative zero, then the result is the same as the argument.

#### Parameters

<i>value</i>	- the floating-point value whose signum is to be returned
--------------	---

#### Returns

the signum function of the argument

#### 6.512.3.23 static float decaf::lang::Math::signum ( float *value* ) [static]

Returns Euler's number  $e$  raised to the power of a double value.

Special cases:

o If the argument is NaN, the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative infinity, then the result is positive zero.

#### Parameters

<i>value</i>	- the exponent to raise $e$ to
--------------	--------------------------------

#### Returns

the value  $e^a$ , where  $e$  is the base of the natural logarithms. Returns  $e^x - 1$ . Note that for values of  $x$  near 0, the exact sum of  $\expm1(x) + 1$  is much closer to the true result of  $e^x$  than  $\exp(x)$ . Special cases:

o If the argument is NaN, the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative infinity, then the result is -1.0. o If the argument is zero, then the result is a zero with the same sign as the argument.

#### Parameters

<i>value</i>	- the value to raise $e^x - 1$
--------------	--------------------------------

#### Returns

the value  $e^x - 1$ . Returns  $\sqrt{x^2 + y^2}$  without intermediate overflow or underflow. Special cases:

If either argument is infinite, then the result is positive infinity. If either argument is NaN and neither argument is infinite, then the result is NaN.

#### Parameters

<i>x</i>	- an argument
<i>y</i>	- another argument

**Returns**

the  $\sqrt{x^2 + y^2}$  without intermediate overflow or underflow Returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero. Special Cases:

o If the argument is NaN, then the result is NaN. o If the argument is positive zero or negative zero, then the result is the same as the argument.

**Parameters**

<i>value</i>	- the floating-point value whose signum is to be returned
--------------	---

**Returns**

the signum function of the argument

**6.512.3.24 static double decaf::lang::Math::sqrt ( double *value* ) [static]**

Returns the arc cosine of an angle, in the range of 0.0 through pi.

Special case:

o If the argument is NaN or its absolute value is greater than 1, then the result is NaN.

**Parameters**

<i>value</i>	- the value to return the arc cosine of.
--------------	--

**Returns**

arc cosine of value in radians. Returns the arc sine of an angle, in the range of -pi/2 through pi/2. Special cases:

o If the argument is NaN or its absolute value is greater than 1, then the result is NaN.

o If the argument is zero, then the result is a zero with the same sign as the argument.

**Parameters**

<i>value</i>	- the value to return the arc cosine of.
--------------	--

**Returns**

arc cosine of value in radians. Returns the arc tangent of an angle, in the range of -pi/2 through pi/2. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

**Parameters**

<i>value</i>	- the value to return the arc cosine of.
--------------	--

**Returns**

arc tangent of value in radians. Converts rectangular coordinates (x, y) to polar (r, theta). This method computes the phase theta by computing an arc tangent of y/x in the range of -pi to pi. Special cases:

o If either argument is NaN, then the result is NaN. o If the first argument is positive zero and the second argument is positive, or the first argument is positive and finite and the second argument is positive infinity, then the result is positive zero. o If the first argument is negative zero and the second argument is positive, or the first argument is negative and finite and the second argument is positive infinity, then the result is negative zero. o If the first argument is positive zero and the second argument is negative, or the first argument is positive and finite and the second argument is negative infinity, then the result is the double value closest to pi. o If the first argument is negative zero and the second argument is negative, or the first argument is negative and finite and the second argument is negative infinity, then the result is the double value closest to -pi. o If the first argument is positive and the second argument is positive zero or negative zero, or the first argument is positive infinity and the second argument is finite, then the result is the double value closest to pi/2. o If the first argument is negative and the second argument is positive zero or negative zero, or the first argument is negative infinity and the second argument is finite, then the result is the double value closest to -pi/2. o If both arguments are positive infinity, then the result is the double value closest to pi/4. o If the first argument is positive infinity and the second argument is negative infinity, then the result is the double value closest to 3\*pi/4. o If the first argument is negative infinity and the second argument is positive infinity, then the result is the double value closest to -pi/4. o If both arguments are negative infinity, then the result is the double value closest to -3\*pi/4.

**Parameters**

y	- the ordinate coordinate
x	- the abscissa coordinate

**Returns**

the theta component of the point (r, theta) in polar coordinates that corresponds to the point (x, y) in Cartesian coordinates. Returns the cube root of a double value. For positive finite x,  $\text{cbrt}(-x) == -\text{cbrt}(x)$ ; that is, the cube root of a negative value is the negative of the cube root of that value's magnitude. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is an infinity with the same sign as the argument. o If the argument is zero, then the result is a zero with the same sign as the argument.

**Parameters**

value	- the double to compute the cube root of
-------	--

**Returns**

the cube root of value Returns the trigonometric cosine of an angle. Special cases:



o If the argument is NaN or an infinity, then the result is NaN.

**Parameters**

<i>value</i>	- an value in radians
--------------	-----------------------

**Returns**

the cosine of the argument. Returns the hyperbolic cosine of a double value. The hyperbolic cosine of  $x$  is defined to be  $(e^x + e^{-x})/2$  where  $e$  is Euler's number. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is positive infinity. o If the argument is zero, then the result is 1.0.

**Parameters**

<i>value</i>	- the number whose hyperbolic cosine is to be found
--------------	---

**Returns**

the hyperbolic cosine of value Returns the trigonometric sine of an angle. Special case:

o If the argument is NaN or an infinity, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

**Parameters**

<i>value</i>	- the number whose sin is to be found
--------------	---------------------------------------

**Returns**

the sine of value Returns the hyperbolic sine of a double value. The hyperbolic sine of  $x$  is defined to be  $(e^x - e^{-x})/2$  where  $e$  is Euler's number. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is an infinity with the same sign as the argument. o If the argument is zero, then the result is a zero with the same sign as the argument.

**Parameters**

<i>value</i>	- the number whose hyperbolic sin is to be found
--------------	--

**Returns**

the hyperbolic sine of value Returns the trigonometric tangent of an angle. Special cases:

o If the argument is NaN or an infinity, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

**Parameters**

<i>value</i>	- the number whose tangent is to be found
--------------	---

**Returns**

the tangent of value Returns the hyperbolic tangent of a double value. The hyperbolic tangent of  $x$  is defined to be  $(e^x - e^{-x}) / (e^x + e^{-x})$ , in other words,  $\sinh(x) / \cosh(x)$ . Note that the absolute value of the exact  $\tanh$  is always less than 1. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument. o If the argument is positive infinity, then the result is +1.0. o If the argument is negative infinity, then the result is -1.0.

**Parameters**

<i>value</i>	- the number whose hyperbolic tangent is to be found
--------------	--

**Returns**

the hyperbolic cosine of value Returns the correctly rounded positive square root of a double value. Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is the same as the argument.

Otherwise, the result is the double value closest to the true mathematical square root of the argument value.

**Parameters**

<i>value</i>	- the value to find the square root of
<i>the</i>	square root of the argument.

**6.512.3.25** `static double decaf::lang::Math::toDegrees ( double angrad )` [`inline`, `static`]

Returns the measure in degrees of the supplied radian angle.

**Parameters**

<i>angrad</i>	- an angle in radians
---------------	-----------------------

**Returns**

the degree measure of the angle.

**6.512.3.26** `static double decaf::lang::Math::toRadians ( double angdeg )` [`inline`, `static`]

Returns the measure in radians of the supplied degree angle.

#### Parameters

<i>angdeg</i>	- an angle in degrees
---------------	-----------------------

#### Returns

the radian measure of the angle.

### 6.512.4 Field Documentation

**6.512.4.1** `const double decaf::lang::Math::E` [`static`]

**6.512.4.2** `const double decaf::lang::Math::PI` [`static`]

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Math.h`

## 6.513 activemq::util::MemoryUsage Class Reference

```
#include <src/main/activemq/util/MemoryUsage.h>
```

Inheritance diagram for `activemq::util::MemoryUsage`:

### Public Member Functions

- **MemoryUsage ()**  
*Default Constructor.*
- **MemoryUsage (unsigned long long limit)**  
*Creates an instance of an **Usage** (p. 4075) monitor with a set limit.*
- **virtual ~MemoryUsage ()**
- **virtual void waitForSpace ()**  
*Waits forever for more space to be returned to this **Usage** (p. 4075) Manager.*
- **virtual void waitForSpace (unsigned int timeout)**  
*Waits for more space to be returned to this **Usage** (p. 4075) Manager, times out when the given time span in milliseconds elapses.*

- virtual void **enqueueUsage** (unsigned long long value)  
*Tries to increase the usage by value amount but blocks if this object is currently full.*
- virtual void **increaseUsage** (unsigned long long value)  
*Increases the usage by the value amount.*
- virtual void **decreaseUsage** (unsigned long long value)  
*Decreases the usage by the value amount.*
- virtual bool **isFull** () const  
*Returns true if this **Usage** (p. 4075) instance is full, i.e.*
- unsigned long long **getUsage** () const  
*Gets the current usage amount.*
- void **setUsage** (unsigned long long usage)  
*Sets the current usage amount.*
- unsigned long long **getLimit** () const  
*Gets the current limit amount.*
- void **setLimit** (unsigned long long limit)  
*Sets the current limit amount.*

### 6.513.1 Constructor & Destructor Documentation

#### 6.513.1.1 activemq::util::MemoryUsage::MemoryUsage ( )

Default Constructor.

#### 6.513.1.2 activemq::util::MemoryUsage::MemoryUsage ( unsigned long long *limit* )

Creates an instance of an **Usage** (p. 4075) monitor with a set limit.

#### Parameters

<i>limit</i>	- amount of memory this manager allows.
--------------	---

6.513.1.3 `virtual activemq::util::MemoryUsage::~~MemoryUsage ( ) [virtual]`

## 6.513.2 Member Function Documentation

6.513.2.1 `virtual void activemq::util::MemoryUsage::decreaseUsage ( unsigned long long value ) [virtual]`

Decreases the usage by the value amount.

### Parameters

<i>value</i>	Amount of space to return to the pool
--------------	---------------------------------------

Implements `activemq::util::Usage` (p. 4076).

6.513.2.2 `virtual void activemq::util::MemoryUsage::enqueueUsage ( unsigned long long value ) [inline, virtual]`

Tries to increase the usage by value amount but blocks if this object is currently full.

### Parameters

<i>value</i>	Amount of usage in bytes to add.
--------------	----------------------------------

Implements `activemq::util::Usage` (p. 4076).

6.513.2.3 `unsigned long long activemq::util::MemoryUsage::getLimit ( ) const [inline]`

Gets the current limit amount.

### Returns

the amount that can be used before full.

6.513.2.4 `unsigned long long activemq::util::MemoryUsage::getUsage ( ) const [inline]`

Gets the current usage amount.

### Returns

the amount of bytes currently used.

6.513.2.5 `virtual void activemq::util::MemoryUsage::increaseUsage ( unsigned long long value ) [virtual]`

Increases the usage by the value amount.

**Parameters**

<i>value</i>	Amount of usage to add.
--------------	-------------------------

Implements **activemq::util::Usage** (p. 4077).

**6.513.2.6** `virtual bool activemq::util::MemoryUsage::isFull ( ) const` `[virtual]`

Returns true if this **Usage** (p. 4075) instance is full, i.e.

**Usage** (p. 4075)  $\geq 100\%$

Implements **activemq::util::Usage** (p. 4077).

**6.513.2.7** `void activemq::util::MemoryUsage::setLimit ( unsigned long long limit )`  
`[inline]`

Sets the current limit amount.

**Parameters**

<i>limit</i>	- The amount that can be used before full.
--------------	--

**6.513.2.8** `void activemq::util::MemoryUsage::setUsage ( unsigned long long usage )`  
`[inline]`

Sets the current usage amount.

**Parameters**

<i>usage</i>	- The amount to tag as used.
--------------	------------------------------

**6.513.2.9** `virtual void activemq::util::MemoryUsage::waitForSpace ( )` `[virtual]`

Waits forever for more space to be returned to this **Usage** (p. 4075) Manager.

Implements **activemq::util::Usage** (p. 4077).

**6.513.2.10** `virtual void activemq::util::MemoryUsage::waitForSpace ( unsigned int timeout )`  
`[virtual]`

Waits for more space to be returned to this **Usage** (p. 4075) Manager, times out when the given time span in milliseconds elapses.

**Parameters**

<i>timeout</i>	The time to wait for more space.
----------------	----------------------------------

Implements **activemq::util::Usage** (p. 4077).

The documentation for this class was generated from the following file:

- src/main/activemq/util/**MemoryUsage.h**

## 6.514 activemq::commands::Message Class Reference

```
#include <src/main/activemq/commands/Message.h>
```

Inheritance diagram for **activemq::commands::Message**:

### Public Member Functions

- **Message** ()
- virtual **~Message** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **Message \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual void **beforeMarshal** (**wireformat::WireFormat** \*wireFormat AMQCPP\_UNUSED) throw ( decaf::io::IOException )  
*Handles the marshaling of the objects properties into the internal byte array before the object is marshaled to the wire.*
- virtual void **afterUnmarshal** (**wireformat::WireFormat** \*wireFormat AMQCPP\_UNUSED) throw ( decaf::io::IOException )  
*Called after unmarshaling is started to cleanup the object being unmarshaled.*
- virtual bool **isMarshalAware** () const

*Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.*

- virtual void **setAckHandler** (const **Pointer**< **core::ActiveMQAckHandler** > &handler)  
*Sets the Acknowledgment Handler that this **Message** (p. 2596) will use when the Acknowledge method is called.*
- virtual **Pointer**< **core::ActiveMQAckHandler** > **getAckHandler** () const  
*Gets the Acknowledgment Handler that this **Message** (p. 2596) will use when the Acknowledge method is called.*
- void **setConnection** (**core::ActiveMQConnection** \*connection)  
*Sets the ActiveMQConnection instance that this **Command** (p. 1227) was created from when the session create methods are called to create a **Message** (p. 2596).*
- **core::ActiveMQConnection** \* **getConnection** () const  
*Gets the ActiveMQConnection instance that this **Command** (p. 1227) was created from when the session create methods are called to create a **Message** (p. 2596).*
- virtual unsigned int **getSize** () const  
*Returns the Size of this message in Bytes.*
- virtual bool **isExpired** () const  
*Returns if this message has expired, meaning that its Expiration time has elapsed.*
- virtual void **onSend** ()  
*Allows derived **Message** (p. 2596) classes to perform tasks before a message is sent.*
- **util::PrimitiveMap** & **getMessageProperties** ()  
*Gets a reference to the Message's Properties object, allows the derived classes to get and set their own specific properties.*
- const **util::PrimitiveMap** & **getMessageProperties** () const
- bool **isReadOnlyProperties** () const  
*Returns if the **Message** (p. 2596) Properties Are Read Only.*
- void **setReadOnlyProperties** (bool value)  
*Set the Read Only State of the **Message** (p. 2596) Properties.*
- bool **isReadOnlyBody** () const  
*Returns if the **Message** (p. 2596) Body is Read Only.*
- void **setReadOnlyBody** (bool value)  
*Set the Read Only State of the **Message** (p. 2596) Content.*
- virtual const **Pointer**< **ProducerId** > & **getProducerId** () const



- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &**producerId**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &**destination**)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &**transactionId**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getOriginalDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getOriginalDestination** ()
- virtual void **setOriginalDestination** (const **Pointer**< **ActiveMQDestination** > &**originalDestination**)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &**messageId**)
- virtual const **Pointer**< **TransactionId** > & **getOriginalTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getOriginalTransactionId** ()
- virtual void **setOriginalTransactionId** (const **Pointer**< **TransactionId** > &**originalTransactionId**)
- virtual const std::string & **getGroupID** () const
- virtual std::string & **getGroupID** ()
- virtual void **setGroupID** (const std::string &**groupID**)
- virtual int **getGroupSequence** () const
- virtual void **setGroupSequence** (int **groupSequence**)
- virtual const std::string & **getCorrelationId** () const
- virtual std::string & **getCorrelationId** ()
- virtual void **setCorrelationId** (const std::string &**correlationId**)
- virtual bool **isPersistent** () const
- virtual void **setPersistent** (bool **persistent**)
- virtual long long **getExpiration** () const
- virtual void **setExpiration** (long long **expiration**)
- virtual unsigned char **getPriority** () const
- virtual void **setPriority** (unsigned char **priority**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getReplyTo** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getReplyTo** ()
- virtual void **setReplyTo** (const **Pointer**< **ActiveMQDestination** > &**replyTo**)
- virtual long long **getTimestamp** () const
- virtual void **setTimestamp** (long long **timestamp**)
- virtual const std::string & **getType** () const
- virtual std::string & **getType** ()
- virtual void **setType** (const std::string &**type**)
- virtual const std::vector< unsigned char > & **getContent** () const
- virtual std::vector< unsigned char > & **getContent** ()
- virtual void **setContent** (const std::vector< unsigned char > &**content**)

- virtual const std::vector< unsigned char > & **getMarshaledProperties** () const
- virtual std::vector< unsigned char > & **getMarshaledProperties** ()
- virtual void **setMarshaledProperties** (const std::vector< unsigned char > &**marshalledProperties**)
- virtual const **Pointer**< **DataStructure** > & **getDataStructure** () const
- virtual **Pointer**< **DataStructure** > & **getDataStructure** ()
- virtual void **setDataStructure** (const **Pointer**< **DataStructure** > &**dataStructure**)
- virtual const **Pointer**< **ConsumerId** > & **getTargetConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getTargetConsumerId** ()
- virtual void **setTargetConsumerId** (const **Pointer**< **ConsumerId** > &**targetConsumerId**)
- virtual bool **isCompressed** () const
- virtual void **setCompressed** (bool **compressed**)
- virtual int **getRedeliveryCounter** () const
- virtual void **setRedeliveryCounter** (int **redeliveryCounter**)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &**brokerPath**)
- virtual long long **getArrival** () const
- virtual void **setArrival** (long long **arrival**)
- virtual const std::string & **getUserID** () const
- virtual std::string & **getUserID** ()
- virtual void **setUserID** (const std::string &**userID**)
- virtual bool **isRecievedByDFBridge** () const
- virtual void **setRecievedByDFBridge** (bool **recievedByDFBridge**)
- virtual bool **isDroppable** () const
- virtual void **setDroppable** (bool **droppable**)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getCluster** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getCluster** ()
- virtual void **setCluster** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &**cluster**)
- virtual long long **getBrokerInTime** () const
- virtual void **setBrokerInTime** (long long **brokerInTime**)
- virtual long long **getBrokerOutTime** () const
- virtual void **setBrokerOutTime** (long long **brokerOutTime**)
- virtual bool **isMessage** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** \*visitor) throw ( **exceptions::ActiveMQException** )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_MESSAGE** = 0

### Protected Attributes

- **core::ActiveMQConnection \* connection**
- **Pointer< ProducerId > producerId**
- **Pointer< ActiveMQDestination > destination**
- **Pointer< TransactionId > transactionId**
- **Pointer< ActiveMQDestination > originalDestination**
- **Pointer< MessageId > messageId**
- **Pointer< TransactionId > originalTransactionId**
- std::string **groupId**
- int **groupSequence**
- std::string **correlationId**
- bool **persistent**
- long long **expiration**
- unsigned char **priority**
- **Pointer< ActiveMQDestination > replyTo**
- long long **timestamp**
- std::string **type**
- std::vector< unsigned char > **content**
- std::vector< unsigned char > **marshalledProperties**
- **Pointer< DataStructure > dataStructure**
- **Pointer< ConsumerId > targetConsumerId**
- bool **compressed**
- int **redeliveryCounter**
- std::vector< decaf::lang::Pointer< BrokerId > > **brokerPath**
- long long **arrival**
- std::string **userID**
- bool **recievedByDFBridge**
- bool **droppable**
- std::vector< decaf::lang::Pointer< BrokerId > > **cluster**
- long long **brokerInTime**
- long long **brokerOutTime**

### Static Protected Attributes

- static const unsigned int **DEFAULT\_MESSAGE\_SIZE** = 1024

### 6.514.1 Constructor & Destructor Documentation

6.514.1.1 `activemq::commands::Message::Message ( )`

6.514.1.2 `virtual activemq::commands::Message::~~Message ( ) [virtual]`

### 6.514.2 Member Function Documentation

6.514.2.1 `virtual void activemq::commands::Message::afterUnmarshal ( wireformat::WireFormat *wireFormat AMQCPP_UNUSED ) throw ( decaf::io::IOException ) [virtual]`

Called after unmarshaling is started to cleanup the object being unmarshaled.

#### Parameters

<i>wireFormat</i>	- the wireformat object to control unmarshaling
-------------------	---

Reimplemented from `activemq::commands::BaseDataStructure` (p. 839).

6.514.2.2 `virtual void activemq::commands::Message::beforeMarshal ( wireformat::WireFormat *wireFormat AMQCPP_UNUSED ) throw ( decaf::io::IOException ) [virtual]`

Handles the marshaling of the objects properties into the internal byte array before the object is marshaled to the wire.

#### Parameters

<i>wireFormat</i>	- the wireformat controller
-------------------	-----------------------------

Reimplemented from `activemq::commands::BaseDataStructure` (p. 839).

6.514.2.3 `virtual Message* activemq::commands::Message::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1714).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 186), `activemq::commands::ActiveMQTextMessage` (p. 219), `activemq::commands::ActiveMQMapMessage` (p. 354), `activemq::commands::ActiveMQMessage` (p. 391), `activemq::commands::ActiveMQObjectMessage` (p. 439), `activemq::commands::ActiveMQStreamMessage` (p. 539), and `activemq::commands::ActiveMQTextMessage` (p. 669).

**6.514.2.4** `virtual void activemq::commands::Message::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 186), `activemq::commands::ActiveMQBytesMessage` (p. 219), `activemq::commands::ActiveMQMapMessage` (p. 354), `activemq::commands::ActiveMQMessage` (p. 391), `activemq::commands::ActiveMQObjectMessage` (p. 439), `activemq::commands::ActiveMQStreamMessage` (p. 540), and `activemq::commands::ActiveMQTextMessage` (p. 669).

**6.514.2.5** `virtual bool activemq::commands::Message::equals ( const DataStructure * value ) const [virtual]`

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 187), `activemq::commands::ActiveMQBytesMessage` (p. 220), `activemq::commands::ActiveMQMapMessage` (p. 355), `activemq::commands::ActiveMQMessage` (p. 392), `activemq::commands::ActiveMQMessageTemplate< T >` (p. 423), `activemq::commands::ActiveMQObjectMessage` (p. 439), `activemq::commands::ActiveMQStreamMessage` (p. 540), `activemq::commands::ActiveMQTextMessage` (p. 669), `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 423), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 423), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 423), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 423), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 423), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 423).

**6.514.2.6** `virtual Pointer<core::ActiveMQAckHandler> activemq::commands::Message::getAckHandler ( ) const [inline, virtual]`

Gets the Acknowledgment Handler that this `Message` (p. 2596) will use when the Acknowledge method is called.

**Returns**

handler ActiveMQAckHandler to call or NULL if not set

6.514.2.7 `virtual long long activemq::commands::Message::getArrival ( ) const`  
[virtual]

6.514.2.8 `virtual long long activemq::commands::Message::getBrokerInTime ( ) const`  
[virtual]

6.514.2.9 `virtual long long activemq::commands::Message::getBrokerOutTime ( ) const`  
[virtual]

6.514.2.10 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >&`  
`activemq::commands::Message::getBrokerPath ( ) const` [virtual]

6.514.2.11 `virtual std::vector< decaf::lang::Pointer<BrokerId> >&`  
`activemq::commands::Message::getBrokerPath ( )` [virtual]

6.514.2.12 `virtual std::vector< decaf::lang::Pointer<BrokerId> >&`  
`activemq::commands::Message::getCluster ( )` [virtual]

6.514.2.13 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >&`  
`activemq::commands::Message::getCluster ( ) const` [virtual]

6.514.2.14 `core::ActiveMQConnection* activemq::commands::Message::getConnection (`  
`) const` [inline]

Gets the ActiveMQConnection instance that this **Command** (p. 1227) was created from when the session create methods are called to create a **Message** (p. 2596).

**Returns**

the ActiveMQConnection parent for this **Message** (p. 2596) or NULL if not set.

- 6.514.2.15 `virtual const std::vector<unsigned char>& activemq::commands::Message::getContent ( ) const` [virtual]
- 6.514.2.16 `virtual std::vector<unsigned char>& activemq::commands::Message::getContent ( )` [virtual]
- 6.514.2.17 `virtual const std::string& activemq::commands::Message::getCorrelationId ( ) const` [virtual]
- 6.514.2.18 `virtual std::string& activemq::commands::Message::getCorrelationId ( )` [virtual]
- 6.514.2.19 `virtual Pointer<DataStructure>& activemq::commands::Message::getDataStructure ( )` [virtual]
- 6.514.2.20 `virtual const Pointer<DataStructure>& activemq::commands::Message::getDataStructure ( ) const` [virtual]
- 6.514.2.21 `virtual unsigned char activemq::commands::Message::getDataStructureType ( ) const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

### Returns

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 187), **activemq::commands::ActiveMQBytesMessage** (p. 221), **activemq::commands::ActiveMQMapMessage** (p. 356), **activemq::commands::ActiveMQMessage** (p. 392), **activemq::commands::ActiveMQObjectMessage** (p. 440), **activemq::commands::ActiveMQStreamMessage** (p. 540), and **activemq::commands::ActiveMQTextMessage** (p. 669).

- 6.514.2.22 **virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getDestination ( ) const** [virtual]
- 6.514.2.23 **virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getDestination ( )** [virtual]
- 6.514.2.24 **virtual long long activemq::commands::Message::getExpiration ( ) const** [virtual]
- 6.514.2.25 **virtual const std::string& activemq::commands::Message::getGroupID ( ) const** [virtual]
- 6.514.2.26 **virtual std::string& activemq::commands::Message::getGroupID ( )** [virtual]
- 6.514.2.27 **virtual int activemq::commands::Message::getGroupSequence ( ) const** [virtual]
- 6.514.2.28 **virtual const std::vector<unsigned char>& activemq::commands::Message::getMarshaledProperties ( ) const** [virtual]
- 6.514.2.29 **virtual std::vector<unsigned char>& activemq::commands::Message::getMarshaledProperties ( )** [virtual]
- 6.514.2.30 **virtual const Pointer<MessageId>& activemq::commands::Message::getMessageld ( ) const** [virtual]
- 6.514.2.31 **virtual Pointer<MessageId>& activemq::commands::Message::getMessageld ( )** [virtual]
- 6.514.2.32 **util::PrimitiveMap& activemq::commands::Message::getMessageProperties ( )** [inline]

Gets a reference to the Message's Properties object, allows the derived classes to get and set their own specific properties.

### Returns

a reference to the Primitive Map that holds message properties.



- 6.514.2.33 `const util::PrimitiveMap& activemq::commands::Message::getMessageProperties ( ) const`  
[inline]
- 6.514.2.34 `virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getOriginalDestination ( )` [virtual]
- 6.514.2.35 `virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getOriginalDestination ( ) const` [virtual]
- 6.514.2.36 `virtual const Pointer<TransactionId>& activemq::commands::Message::getOriginalTransactionId ( ) const`  
[virtual]
- 6.514.2.37 `virtual Pointer<TransactionId>& activemq::commands::Message::getOriginalTransactionId ( )`  
[virtual]
- 6.514.2.38 `virtual unsigned char activemq::commands::Message::getPriority ( ) const`  
[virtual]
- 6.514.2.39 `virtual const Pointer<ProducerId>& activemq::commands::Message::getProducerId ( ) const`  
[virtual]
- 6.514.2.40 `virtual Pointer<ProducerId>& activemq::commands::Message::getProducerId ( )` [virtual]
- 6.514.2.41 `virtual int activemq::commands::Message::getRedeliveryCounter ( ) const`  
[virtual]
- 6.514.2.42 `virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getReplyTo ( ) const` [virtual]
- 6.514.2.43 `virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getReplyTo ( )` [virtual]
- 6.514.2.44 `virtual unsigned int activemq::commands::Message::getSize ( ) const`  
[virtual]

Returns the Size of this message in Bytes.

### Returns

number of bytes this message equates to.

Reimplemented in `activemq::commands::ActiveMQTextMessage` (p. 670).

- 6.514.2.45 **virtual const Pointer<ConsumerId>& activemq::commands::Message::getTargetConsumerId ( ) const**  
[virtual]
- 6.514.2.46 **virtual Pointer<ConsumerId>& activemq::commands::Message::getTargetConsumerId ( )**  
[virtual]
- 6.514.2.47 **virtual long long activemq::commands::Message::getTimestamp ( ) const**  
[virtual]
- 6.514.2.48 **virtual const Pointer<TransactionId>& activemq::commands::Message::getTransactionId ( ) const** [virtual]
- 6.514.2.49 **virtual Pointer<TransactionId>& activemq::commands::Message::getTransactionId ( )**  
[virtual]
- 6.514.2.50 **virtual const std::string& activemq::commands::Message::getType ( ) const**  
[virtual]
- 6.514.2.51 **virtual std::string& activemq::commands::Message::getType ( )** [virtual]
- 6.514.2.52 **virtual const std::string& activemq::commands::Message::getUserID ( ) const**  
[virtual]
- 6.514.2.53 **virtual std::string& activemq::commands::Message::getUserID ( )** [virtual]
- 6.514.2.54 **virtual bool activemq::commands::Message::isCompressed ( ) const**  
[virtual]
- 6.514.2.55 **virtual bool activemq::commands::Message::isDroppable ( ) const** [virtual]
- 6.514.2.56 **virtual bool activemq::commands::Message::isExpired ( ) const** [virtual]

Returns if this message has expired, meaning that its Expiration time has elapsed.

#### Returns

true if message is expired.

- 6.514.2.57 **virtual bool activemq::commands::Message::isMarshalAware ( ) const**  
[inline, virtual]

Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.

#### Returns

boolean indicating desire to be in marshaling stages

Reimplemented from **activemq::commands::BaseDataStructure** (p. 840).

Reimplemented in **activemq::commands::ActiveMQMapMessage** (p. 359).

**6.514.2.58** `virtual bool activemq::commands::Message::isMessage ( ) const [inline, virtual]`

#### Returns

an answer of true to the **isMessage()** (p. 2608) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 768).

**6.514.2.59** `virtual bool activemq::commands::Message::isPersistent ( ) const [virtual]`

**6.514.2.60** `bool activemq::commands::Message::isReadOnlyBody ( ) const [inline]`

Returns if the **Message** (p. 2596) Body is Read Only.

#### Returns

true if **Message** (p. 2596) Content is Read Only.

**6.514.2.61** `bool activemq::commands::Message::isReadOnlyProperties ( ) const [inline]`

Returns if the **Message** (p. 2596) Properties Are Read Only.

#### Returns

true if **Message** (p. 2596) Properties are Read Only.

**6.514.2.62** `virtual bool activemq::commands::Message::isRecievedByDFBridge ( ) const [virtual]`

**6.514.2.63** `virtual void activemq::commands::Message::onSend ( ) [inline, virtual]`

Allows derived **Message** (p. 2596) classes to perform tasks before a message is sent.

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 221), **activemq::commands::ActiveMQMessageTemplate< T >** (p. 431), **activemq::commands::ActiveMQStreamMessage** (p. 540), **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 431), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 431), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 431), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 431), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 431), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 431).

6.514.2.64 `virtual void activemq::commands::Message::setAckHandler ( const Pointer< core::ActiveMQAckHandler > & handler ) [inline, virtual]`

Sets the Acknowledgment Handler that this **Message** (p. 2596) will use when the Acknowledge method is called.

#### Parameters

<i>handler</i>	ActiveMQAckHandler to call
----------------	----------------------------

6.514.2.65 `virtual void activemq::commands::Message::setArrival ( long long arrival ) [virtual]`

6.514.2.66 `virtual void activemq::commands::Message::setBrokerInTime ( long long brokerInTime ) [virtual]`

6.514.2.67 `virtual void activemq::commands::Message::setBrokerOutTime ( long long brokerOutTime ) [virtual]`

6.514.2.68 `virtual void activemq::commands::Message::setBrokerPath ( const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath ) [virtual]`

6.514.2.69 `virtual void activemq::commands::Message::setCluster ( const std::vector< decaf::lang::Pointer< BrokerId > > & cluster ) [virtual]`

6.514.2.70 `virtual void activemq::commands::Message::setCompressed ( bool compressed ) [virtual]`

6.514.2.71 `void activemq::commands::Message::setConnection ( core::ActiveMQConnection * connection ) [inline]`

Sets the ActiveMQConnection instance that this **Command** (p. 1227) was created from when the session create methods are called to create a **Message** (p. 2596).

#### Parameters

<i>handler</i>	ActiveMQConnection parent for this message
----------------	--

- 6.514.2.72 `virtual void activemq::commands::Message::setContent ( const std::vector< unsigned char > & content ) [virtual]`
- 6.514.2.73 `virtual void activemq::commands::Message::setCorrelationId ( const std::string & correlationId ) [virtual]`
- 6.514.2.74 `virtual void activemq::commands::Message::setDataStructure ( const Pointer< DataStructure > & dataStructure ) [virtual]`
- 6.514.2.75 `virtual void activemq::commands::Message::setDestination ( const Pointer< ActiveMQDestination > & destination ) [virtual]`
- 6.514.2.76 `virtual void activemq::commands::Message::setDroppable ( bool droppable ) [virtual]`
- 6.514.2.77 `virtual void activemq::commands::Message::setExpiration ( long long expiration ) [virtual]`
- 6.514.2.78 `virtual void activemq::commands::Message::setGroupId ( const std::string & groupId ) [virtual]`
- 6.514.2.79 `virtual void activemq::commands::Message::setGroupSequence ( int groupSequence ) [virtual]`
- 6.514.2.80 `virtual void activemq::commands::Message::setMarshaledProperties ( const std::vector< unsigned char > & marshalledProperties ) [virtual]`
- 6.514.2.81 `virtual void activemq::commands::Message::setMessageId ( const Pointer< MessageId > & messageId ) [virtual]`
- 6.514.2.82 `virtual void activemq::commands::Message::setOriginalDestination ( const Pointer< ActiveMQDestination > & originalDestination ) [virtual]`
- 6.514.2.83 `virtual void activemq::commands::Message::setOriginalTransactionId ( const Pointer< TransactionId > & originalTransactionId ) [virtual]`
- 6.514.2.84 `virtual void activemq::commands::Message::setPersistent ( bool persistent ) [virtual]`
- 6.514.2.85 `virtual void activemq::commands::Message::setPriority ( unsigned char priority ) [virtual]`
- 6.514.2.86 `virtual void activemq::commands::Message::setProducerId ( const Pointer< ProducerId > & producerId ) [virtual]`
- 6.514.2.87 `void activemq::commands::Message::setReadOnlyBody ( bool value ) [inline]`

Set the Read Only State of the **Message** (p. 2596) Content.

**Parameters**

<i>value</i>	- true if Content should be read only.
--------------	--

**6.514.2.88** `void activemq::commands::Message::setReadOnlyProperties ( bool value )`  
`[inline]`

Set the Read Only State of the **Message** (p. 2596) Properties.

**Parameters**

<i>value</i>	- true if Properties should be read only.
--------------	---

**6.514.2.89** `virtual void activemq::commands::Message::setRecievedByDFBridge ( bool recievedByDFBridge )`  
`[virtual]`

**6.514.2.90** `virtual void activemq::commands::Message::setRedeliveryCounter ( int redeliveryCounter )`  
`[virtual]`

**6.514.2.91** `virtual void activemq::commands::Message::setReplyTo ( const Pointer< ActiveMQDestination > & replyTo )`  
`[virtual]`

**6.514.2.92** `virtual void activemq::commands::Message::setTargetConsumerId ( const Pointer< ConsumerId > & targetConsumerId )`  
`[virtual]`

**6.514.2.93** `virtual void activemq::commands::Message::setTimestamp ( long long timestamp )`  
`[virtual]`

**6.514.2.94** `virtual void activemq::commands::Message::setTransactionId ( const Pointer< TransactionId > & transactionId )`  
`[virtual]`

**6.514.2.95** `virtual void activemq::commands::Message::setType ( const std::string & type )`  
`[virtual]`

**6.514.2.96** `virtual void activemq::commands::Message::setUserID ( const std::string & userID )`  
`[virtual]`

**6.514.2.97** `virtual std::string activemq::commands::Message::toString ( ) const`  
`[virtual]`

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

**Returns**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 189), `activemq::commands::ActiveMQBytesMessage` (p. 229), `activemq::commands::ActiveMQMapMessage` (p. 364), `activemq::commands::ActiveMQMessage` (p. 392), `activemq::commands::ActiveMQObjectMessage` (p. 440), `activemq::commands::ActiveMQStreamMessage` (p. 547), and `activemq::commands::ActiveMQTextMessage` (p. 671).

```
6.514.2.98 virtual Pointer<Command> activemq::commands::Message::visit  
          ( activemq::state::CommandVisitor * visitor ) throw (   
            exceptions::ActiveMQException ) [virtual]
```

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

### Returns

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 1232).

**6.514.3 Field Documentation**

- 6.514.3.1** `long long activemq::commands::Message::arrival` [protected]
- 6.514.3.2** `long long activemq::commands::Message::brokerInTime`  
[protected]
- 6.514.3.3** `long long activemq::commands::Message::brokerOutTime`  
[protected]
- 6.514.3.4** `std::vector< decaf::lang::Pointer<BrokerId> >`  
`activemq::commands::Message::brokerPath` [protected]
- 6.514.3.5** `std::vector< decaf::lang::Pointer<BrokerId> >`  
`activemq::commands::Message::cluster` [protected]
- 6.514.3.6** `bool activemq::commands::Message::compressed` [protected]
- 6.514.3.7** `core::ActiveMQConnection* ac-`  
`tivemq::commands::Message::connection`  
[protected]
- 6.514.3.8** `std::vector<unsigned char> activemq::commands::Message::content`  
[protected]
- 6.514.3.9** `std::string activemq::commands::Message::correlationId`  
[protected]
- 6.514.3.10** `Pointer<DataStructure> ac-`  
`tivemq::commands::Message::dataStructure`  
[protected]
- 6.514.3.11** `const unsigned int activemq::commands::Message::DEFAULT_-`  
`MESSAGE_SIZE = 1024` [static, protected]
- 6.514.3.12** `Pointer<ActiveMQDestination> ac-`  
`tivemq::commands::Message::destination`  
[protected]
- 6.514.3.13** `bool activemq::commands::Message::droppable` [protected]
- 6.514.3.14** `long long activemq::commands::Message::expiration` [protected]
- 6.514.3.15** `std::string activemq::commands::Message::groupId` [protected]
- 6.514.3.16** `int activemq::commands::Message::groupSequence` [protected]
- 6.514.3.17** `const unsigned char activemq::commands::Message::ID_MESSAGE = 0`  
[static]
- 6.514.3.18** `std::vector<unsigned char> ac-`  
`tivemq::commands::Message::marshalledProperties`  
[protected]
- 6.514.3.19** `Pointer<MessageId> activemq::commands::Message::messageId`  
[protected]
- 6.514.3.20** `Pointer<ActiveMQDestination> ac-`  
`tivemq::commands::Message::originalDestination`



- src/main/activemq/commands/Message.h

## 6.515 cms::Message Class Reference

Root of all messages.

```
#include <src/main/cms/Message.h>
```

Inheritance diagram for cms::Message:

### Public Member Functions

- virtual **~Message** ()
- virtual **Message \* clone** () const =0  
*Clone this message exactly, returns a new instance that the caller is required to delete.*
- virtual void **acknowledge** () const =0 throw ( IllegalStateException, CMSException )  
*Acknowledges all consumed messages of the session of this consumed message.*
- virtual void **clearBody** ()=0 throw ( CMSException )  
*Clears out the body of the message.*
- virtual void **clearProperties** ()=0 throw ( CMSException )  
*Clears out the message body.*
- virtual std::vector< std::string > **getPropertyNames** () const =0 throw ( CMSException )  
*Retrieves the property names.*
- virtual bool **propertyExists** (const std::string &name) const =0 throw ( CMSException )  
*Indicates whether or not a given property exists.*
- virtual bool **getBooleanProperty** (const std::string &name) const =0 throw ( MessageFormatException, CMSException )  
*Gets a boolean property.*
- virtual unsigned char **getByteProperty** (const std::string &name) const =0 throw ( MessageFormatException, CMSException )  
*Gets a byte property.*
- virtual double **getDoubleProperty** (const std::string &name) const =0 throw ( MessageFormatException, CMSException )

*Gets a double property.*

- virtual float **getFloatProperty** (const std::string &name) const =0 throw ( MessageFormatException, CMSEException )

*Gets a float property.*

- virtual int **getIntProperty** (const std::string &name) const =0 throw ( MessageFormatException, CMSEException )

*Gets a int property.*

- virtual long long **getLongProperty** (const std::string &name) const =0 throw ( MessageFormatException, CMSEException )

*Gets a long property.*

- virtual short **getShortProperty** (const std::string &name) const =0 throw ( MessageFormatException, CMSEException )

*Gets a short property.*

- virtual std::string **getStringProperty** (const std::string &name) const =0 throw ( MessageFormatException, CMSEException )

*Gets a string property.*

- virtual void **setBooleanProperty** (const std::string &name, bool value)=0 throw ( MessageNotWriteableException, CMSEException )

*Sets a boolean property.*

- virtual void **setByteProperty** (const std::string &name, unsigned char value)=0 throw ( MessageNotWriteableException, CMSEException )

*Sets a byte property.*

- virtual void **setDoubleProperty** (const std::string &name, double value)=0 throw ( MessageNotWriteableException, CMSEException )

*Sets a double property.*

- virtual void **setFloatProperty** (const std::string &name, float value)=0 throw ( MessageNotWriteableException, CMSEException )

*Sets a float property.*

- virtual void **setIntProperty** (const std::string &name, int value)=0 throw ( MessageNotWriteableException, CMSEException )

*Sets a int property.*

- virtual void **setLongProperty** (const std::string &name, long long value)=0 throw ( MessageNotWriteableException, CMSEException )

*Sets a long property.*

- virtual void **setShortProperty** (const std::string &name, short value)=0 throw ( MessageNotWriteableException, CMSEException )  
*Sets a short property.*
- virtual void **setStringProperty** (const std::string &name, const std::string &value)=0 throw ( MessageNotWriteableException, CMSEException )  
*Sets a string property.*
- virtual std::string **getCMSCorrelationID** () const =0 throw ( CMSEException )  
*Gets the correlation ID for the message.*
- virtual void **setCMSCorrelationID** (const std::string &correlationId)=0 throw ( CMSEException )  
*Sets the correlation ID for the message.*
- virtual int **getCMSDeliveryMode** () const =0 throw ( CMSEException )  
*Gets the **DeliveryMode** (p. 1775) for this message.*
- virtual void **setCMSDeliveryMode** (int mode)=0 throw ( CMSEException )  
*Sets the **DeliveryMode** (p. 1775) for this message.*
- virtual const **Destination** \* **getCMSDestination** () const =0 throw ( CMSEException )  
*Gets the **Destination** (p. 1776) object for this message.*
- virtual void **setCMSDestination** (const **Destination** \*destination)=0 throw ( CMSEException )  
*Sets the **Destination** (p. 1776) object for this message.*
- virtual long long **getCMSExpiration** () const =0 throw ( CMSEException )  
*Gets the message's expiration value.*
- virtual void **setCMSExpiration** (long long expireTime)=0 throw ( CMSEException )  
*Sets the message's expiration value.*
- virtual std::string **getCMSMessageID** () const =0 throw ( CMSEException )  
*The **CMSMessageID** header field contains a value that uniquely identifies each message sent by a provider.*
- virtual void **setCMSMessageID** (const std::string &id)=0 throw ( CMSEException )  
*Sets the message ID.*
- virtual int **getCMSPriority** () const =0 throw ( CMSEException )  
*Gets the message priority level.*

- virtual void **setCMSPriority** (int priority)=0 throw ( CMSEException )  
*Sets the Priority Value for this message.*
- virtual bool **getCMSRedelivered** () const =0 throw ( CMSEException )  
*Gets an indication of whether this message is being redelivered.*
- virtual void **setCMSRedelivered** (bool redelivered)=0 throw ( CMSEException )  
*Specifies whether this message is being redelivered.*
- virtual const **cms::Destination** \* **getCMSReplyTo** () const =0 throw ( CMSEException )  
*Gets the **Destination** (p. 1776) object to which a reply to this message should be sent.*
- virtual void **setCMSReplyTo** (const **cms::Destination** \*destination)=0 throw ( CMSEException )  
*Sets the **Destination** (p. 1776) object to which a reply to this message should be sent.*
- virtual long long **getCMSTimestamp** () const =0 throw ( CMSEException )  
*Gets the message timestamp.*
- virtual void **setCMSTimestamp** (long long timeStamp)=0 throw ( CMSEException )  
*Sets the message timestamp.*
- virtual std::string **getCMSType** () const =0 throw ( CMSEException )  
*Gets the message type identifier supplied by the client when the message was sent.*
- virtual void **setCMSType** (const std::string &type)=0 throw ( CMSEException )  
*Sets the message type.*

### 6.515.1 Detailed Description

Root of all messages. As in JMS, a message is comprised of 3 parts: CMS-specific headers, user-defined properties, and the body.

#### Message (p. 2614) Bodies

The CMS API defines four types of message bodies, each type is contained within its own **Message** (p. 2614) Interface definition.

- Stream - A **StreamMessage** (p. 3760) object's message body contains a stream of primitive values in the C++ language. It is filled and read sequentially. Unlike the **BytesMessage** (p. 1079) type the values written to a **StreamMessage** (p. 3760) retain information on their type and rules for type conversion are enforced when reading back the values from the **Message** (p. 2614) Body.

- Map - A **MapMessage** (p. 2551) object's message body contains a set of name-value pairs, where names are `std::string` objects, and values are C++ primitives. The entries can be accessed sequentially or randomly by name. The **MapMessage** (p. 2551) makes no guarantee on the order of the elements within the **Message** (p. 2614) body.
- Text - A **TextMessage** (p. 3874) object's message body contains a `std::string` object. This message type can be used to transport plain-text messages, and XML messages.
- Bytes - A **BytesMessage** (p. 1079) object's message body contains a stream of uninterpreted bytes. This message type is for literally encoding a body to match an existing message format. In many cases, it is possible to use one of the other body types, which are easier to use.

### Message (p. 2614) Properties

**Message** (p. 2614) properties support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSEException** (p. 1190). The String-to-primitive conversions may throw a runtime exception if the primitive's `valueOf` method does not accept the String as a valid representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	int	long	float	double	String
boolean	X							X
byte		X	X	X	X			X
short			X	X	X			X
int				X	X			X
long					X			X
float						X	X	X
double							X	X
String	X	X	X	X	X	X	X	X

When a **Message** (p. 2614) is delivered its properties are considered to be in a read-only mode and cannot be changed. Attempting to change the value of a delivered **Message**'s properties will result in a **CMSEException** (p. 1190) being thrown.

### See also

JMS API

### Since

1.0

### 6.515.2 Constructor & Destructor Documentation

6.515.2.1 `virtual cms::Message::~~Message ( ) [inline, virtual]`

### 6.515.3 Member Function Documentation

6.515.3.1 `virtual void cms::Message::acknowledge ( ) const throw ( IllegalStateException, CMSEException ) [pure virtual]`

Acknowledges all consumed messages of the session of this consumed message.

All consumed CMS messages support the acknowledge method for use when a client has specified that its CMS session's consumed messages are to be explicitly acknowledged. By invoking acknowledge on a consumed message, a client acknowledges all messages consumed by the session that the message was delivered to.

Calls to acknowledge are ignored for both transacted sessions and sessions specified to use implicit acknowledgment modes.

A client may individually acknowledge each message as it is consumed, or it may choose to acknowledge messages as an application-defined group (which is done by calling acknowledge on the last received message of the group, thereby acknowledging all messages consumed by the session.)

Messages that have been received but not acknowledged may be redelivered.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if an internal error occurs.
<b><i>IllegalStateException</i></b> (p. 2059)	- if this method is called on a closed session.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 422), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 422), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 422), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 422), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 422), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 422).

6.515.3.2 `virtual void cms::Message::clearBody ( ) throw ( CMSEException ) [pure virtual]`

Clears out the body of the message.

This does not clear the headers or properties.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 219), `activemq::commands::ActiveMQMapMessage` (p. 354), `activemq::commands::ActiveMQStreamMessage` (p. 539), `activemq::commands::ActiveMQTextMessage` (p. 668), `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 422), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 422), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 422), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 422), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 422), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 422).

**6.515.3.3** `virtual void cms::Message::clearProperties ( ) throw ( CMSEException )` [pure virtual]

Clears out the message body.

Clearing a message's body does not clear its header values or property entries.

If this message body was read-only, calling this method leaves the message body in the same state as an empty body in a newly created message.

### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 423), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 423), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 423), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 423), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 423), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 423).

**6.515.3.4** `virtual Message* cms::Message::clone ( ) const` [pure virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

### Returns

new copy of this message

Implemented in `activemq::commands::ActiveMQBlobMessage` (p. 186), `activemq::commands::ActiveMQBytesMessage` (p. 219), `activemq::commands::ActiveMQMapMessage` (p. 354), `activemq::commands::ActiveMQMessage` (p. 391), `activemq::commands::ActiveMQObjectMessage` (p. 439), `activemq::commands::ActiveMQStreamMessage` (p. 539), `activemq::commands::ActiveMQTextMessage` (p. 668), and `cms::BytesMessage` (p. 1083).

**6.515.3.5** `virtual bool cms::Message::getBooleanProperty ( const std::string & name ) const  
throw ( MessageFormatException, CMSException ) [pure virtual]`

Gets a boolean property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	if the property does not exist.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 423), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 423), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 423), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 423), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 423), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 423).

**6.515.3.6** `virtual unsigned char cms::Message::getByteProperty ( const std::string & name )  
const throw ( MessageFormatException, CMSException ) [pure  
virtual]`

Gets a byte property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	if the property does not exist.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.



Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 424), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 424), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 424), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 424), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 424), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 424).

**6.515.3.7** `virtual std::string cms::Message::getCMSCorrelationID ( ) const throw ( CMSEException )` [pure virtual]

Gets the correlation ID for the message.

This method is used to return correlation ID values that are either provider-specific message IDs or application-specific String values.

#### Returns

string representation of the correlation Id

#### Exceptions

<i><b>CMSEException</b></i> (p. 1190)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 424), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 424), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 424), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 424), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 424), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 424).

**6.515.3.8** `virtual int cms::Message::getCMSDeliveryMode ( ) const throw ( CMSEException )` [pure virtual]

Gets the **DeliveryMode** (p. 1775) for this message.

#### Returns

**DeliveryMode** (p. 1775) enumerated value.

#### Exceptions

<i><b>CMSEException</b></i> (p. 1190)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 425), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >`

> (p. 425), `activemq::commands::ActiveMQMessageTemplate< cms::Message >`  
 (p. 425), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 425), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 425), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 425).

**6.515.3.9** `virtual const Destination* cms::Message::getCMSDestination ( ) const throw ( CMSException ) [pure virtual]`

Gets the **Destination** (p. 1776) object for this message.

The `CMSDestination` header field contains the destination to which the message is being sent.

When a message is sent, this field is ignored. After completion of the send or publish method, the field holds the destination specified by the method.

When a message is received, its `CMSDestination` value must be equivalent to the value assigned when it was sent.

#### Returns

**Destination** (p. 1776) object

#### Exceptions

<i><b>CMSException</b></i> (p. 1190)	- if an internal error occurs.
---	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage`  
 > (p. 425), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage`  
 > (p. 425), `activemq::commands::ActiveMQMessageTemplate< cms::Message >`  
 (p. 425), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 425), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 425), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 425).

**6.515.3.10** `virtual long long cms::Message::getCMSExpiration ( ) const throw ( CMSException ) [pure virtual]`

Gets the message's expiration value.

When a message is sent, the `CMSExpiration` header field is left unassigned. After completion of the send or publish method, it holds the expiration time of the message. This is the sum of the time-to-live value specified by the client and the GMT at the time of the send or publish.

If the time-to-live is specified as zero, `CMSExpiration` is set to zero to indicate that the message does not expire.

When a message's expiration time is reached, a provider should discard it. The CMS API does not define any form of notification of message expiration.

Clients should not receive messages that have expired; however, the CMS API does not guarantee that this will not happen.

### Returns

the time the message expires, which is the sum of the time-to-live value specified by the client and the GMT at the time of the send

### Exceptions

<i><b>CMSException</b></i> (p. 1190)	- if an internal error occurs.
---	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 425), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 425), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 425), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 425), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 425), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 425).

**6.515.3.11** `virtual std::string cms::Message::getCMSMessageID ( ) const throw ( CMSException ) [pure virtual]`

The CMSMessageID header field contains a value that uniquely identifies each message sent by a provider.

When a message is sent, CMSMessageID can be ignored. When the send or publish method returns, it contains a provider-assigned value.

A CMSMessageID is a String value that should function as a unique key for identifying messages in a historical repository. The exact scope of uniqueness is provider-defined. It should at least cover all messages for a specific installation of a provider, where an installation is some connected set of message routers.

All CMSMessageID values must start with the prefix 'ID:'. Uniqueness of message ID values across different providers is not required.

Since message IDs take some effort to create and increase a message's size, some CMS providers may be able to optimize message overhead if they are given a hint that the message ID is not used by an application. By calling the **MessageProducer.setDisableMessageID** (p. 2816) method, a CMS client enables this potential optimization for all messages sent by that message producer. If the CMS provider accepts this hint, these messages must have the message ID set to null; if the provider ignores the hint, the message ID must be set to its normal unique value.

### Returns

provider-assigned message id

### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 426), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 426), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 426), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 426), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 426), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 426).

**6.515.3.12** `virtual int cms::Message::getCMSPriority ( ) const throw ( CMSEException )`  
[pure virtual]

Gets the message priority level.

The CMS API defines ten levels of priority value, with 0 as the lowest priority and 9 as the highest. In addition, clients should consider priorities 0-4 as gradations of normal priority and priorities 5-9 as gradations of expedited priority.

The CMS API does not require that a provider strictly implement priority ordering of messages; however, it should do its best to deliver expedited messages ahead of normal messages.

#### Returns

priority value

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 426), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 426), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 426), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 426), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 426), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 426).

**6.515.3.13** `virtual bool cms::Message::getCMSRedelivered ( ) const throw ( CMSEException )`  
[pure virtual]

Gets an indication of whether this message is being redelivered.

If a client receives a message with the CMSRedelivered field set, it is likely, but not guaranteed, that this message was delivered earlier but that its receipt was not acknowledged at that time.

**Returns**

true if this message is being redelivered

**Exceptions**

<i>CMSEException</i> (p. 1190)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 426), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 426), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 426), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 426), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 426), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 426).

**6.515.3.14** `virtual const cms::Destination* cms::Message::getCMSReplyTo ( ) const throw ( CMSEException )` [pure virtual]

Gets the **Destination** (p. 1776) object to which a reply to this message should be sent.

**Returns**

**Destination** (p. 1776) to which to send a response to this message

**Exceptions**

<i>CMSEException</i> (p. 1190)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 427), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 427), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 427), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 427), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 427), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 427).

**6.515.3.15** `virtual long long cms::Message::getCMSTimestamp ( ) const throw ( CMSEException )` [pure virtual]

Gets the message timestamp.

The `CMSTimestamp` header field contains the time a message was handed off to a provider to be sent. It is not the time the message was actually transmitted, because the actual send may occur later due to transactions or other client-side queuing of messages.

When a message is sent, `CMSTimestamp` is ignored. When the `send` or `publish` method

returns, it contains a time value somewhere in the interval between the call and the return. The value is in the format of a normal millis time value in the Java programming language.

Since timestamps take some effort to create and increase a message's size, some CMS providers may be able to optimize message overhead if they are given a hint that the timestamp is not used by an application. By calling the `MessageProducer.setDisableMessageTimestamp` method, a CMS client enables this potential optimization for all messages sent by that message producer. If the CMS provider accepts this hint, these messages must have the timestamp set to zero; if the provider ignores the hint, the timestamp must be set to its normal value.

### Returns

the message timestamp

### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 427), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 427), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 427), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 427), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 427), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 427).

**6.515.3.16** `virtual std::string cms::Message::getCMSType ( ) const throw ( CMSEException )`  
[pure virtual]

Gets the message type identifier supplied by the client when the message was sent.

### Returns

the message type

### See also

`setCMSType` (p. 2638)

### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 427), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 427), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 427), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >`

> (p. 427), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 427), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 427).

**6.515.3.17** `virtual double cms::Message::getDoubleProperty ( const std::string & name )  
 const throw ( MessageFormatException, CMSException ) [pure  
 virtual]`

Gets a double property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	if the property does not exist.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage`  
 > (p. 428), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage`  
 > (p. 428), `activemq::commands::ActiveMQMessageTemplate< cms::Message >`  
 (p. 428), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 428), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 428), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 428).

**6.515.3.18** `virtual float cms::Message::getFloatProperty ( const std::string & name ) const throw  
 ( MessageFormatException, CMSException ) [pure virtual]`

Gets a float property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	if the property does not exist.
---	---------------------------------

<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.
---	---------------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 428), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 428), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 428), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 428), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 428), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 428).

**6.515.3.19** `virtual int cms::Message::getIntProperty ( const std::string & name ) const throw ( MessageFormatException, CMSEException ) [pure virtual]`

Gets a int property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	if the property does not exist.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 429), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 429), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 429), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 429), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 429), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 429).

**6.515.3.20** `virtual long long cms::Message::getLongProperty ( const std::string & name ) const throw ( MessageFormatException, CMSEException ) [pure virtual]`

Gets a long property.



**Parameters**

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

**Returns**

The value for the named property.

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	if the property does not exist.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 429), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 429), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 429), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 429), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 429), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 429).

**6.515.3.21** `virtual std::vector<std::string> cms::Message::getPropertyNames ( ) const throw ( CMSException ) [pure virtual]`

Retrieves the property names.

**Returns**

The complete set of property names currently in this message.

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	- if an internal error occurs.
---	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 430), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 430).

**6.515.3.22** `virtual short cms::Message::getShortProperty ( const std::string & name )  
const throw ( MessageFormatException, CMSEException ) [pure  
virtual]`

Gets a short property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	if the property does not exist.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 430), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 430).

**6.515.3.23** `virtual std::string cms::Message::getStringProperty ( const std::string & name ) const throw ( MessageFormatException, CMSEException ) [pure  
virtual]`

Gets a string property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	if the property does not exist.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 430), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 430), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 430).

**6.515.3.24** `virtual bool cms::Message::propertyExists ( const std::string & name ) const throw ( CMSEException ) [pure virtual]`

Indicates whether or not a given property exists.

#### Parameters

<i>name</i>	The name of the property to look up.
-------------	--------------------------------------

#### Returns

True if the property exists in this message.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 431), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 431), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 431), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 431), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 431), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 431).

**6.515.3.25** `virtual void cms::Message::setBooleanProperty ( const std::string & name, bool value ) throw ( MessageNotWriteableException, CMSEException ) [pure virtual]`

Sets a boolean property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if the name is an empty string.
--	-----------------------------------

<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if properties are read-only
---	-------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 431), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 431), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 431), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 431), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 431), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 431).

**6.515.3.26** `virtual void cms::Message::setByteProperty ( const std::string & name, unsigned char value ) throw ( MessageNotWriteableException, CMSEException )`  
[pure virtual]

Sets a byte property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if the name is an empty string.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 432), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 432), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 432), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 432), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 432), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 432).

**6.515.3.27** `virtual void cms::Message::setCMSCorrelationID ( const std::string & correlationId )`  
`throw ( CMSEException )` [pure virtual]

Sets the correlation ID for the message.

A client can use the CMSCorrelationID header field to link one message with another. A typical use is to link a response message with its request message.

CMSCorrelationID can hold one of the following:

- A provider-specific message ID
- An application-specific String
- A provider-native byte[] value

Since each message sent by a CMS provider is assigned a message ID value, it is convenient to link messages via message ID. All message ID values must start with the 'ID:' prefix.

In some cases, an application (made up of several clients) needs to use an application-specific value for linking messages. For instance, an application may use CMSCorrelationID to hold a value referencing some external information. Application-specified values must not start with the 'ID:' prefix; this is reserved for provider-generated message ID values.

If a provider supports the native concept of correlation ID, a CMS client may need to assign specific CMSCorrelationID values to match those expected by clients that do not use the CMS API. A byte[] value is used for this purpose. CMS providers without native correlation ID values are not required to support byte[] values. The use of a byte[] value for CMSCorrelationID is non-portable.

#### Parameters

<i>correlationId</i>	The message ID of a message being referred to.
----------------------	--

#### Exceptions

<b>CMSException</b> (p. 1190)	- if an internal error occurs.
----------------------------------	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 432), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 432), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 432), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 432), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 432), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 432).

**6.515.3.28** `virtual void cms::Message::setCMSDeliveryMode ( int mode ) throw ( CMSException ) [pure virtual]`

Sets the **DeliveryMode** (p. 1775) for this message.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

#### Parameters

<i>mode</i>	<b>DeliveryMode</b> (p. 1775) enumerated value.
-------------	---

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 432), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 432), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 432), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 432), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 432), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 432).

**6.515.3.29** `virtual void cms::Message::setCMSDestination ( const Destination * destination ) throw ( CMSEException ) [pure virtual]`

Sets the **Destination** (p. 1776) object for this message.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

**Parameters**

<i>destination</i>	<b>Destination</b> (p. 1776) Object
--------------------	-------------------------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 433), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 433), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 433), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 433), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 433), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 433).

**6.515.3.30** `virtual void cms::Message::setCMSExpiration ( long long expireTime ) throw ( CMSEException ) [pure virtual]`

Sets the message's expiration value.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

**Parameters**

<i>expireTime</i>	the message's expiration time
-------------------	-------------------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 433), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 433), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 433), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 433), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 433), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 433).

**6.515.3.31** `virtual void cms::Message::setCMSMessageID ( const std::string & id ) throw ( CMSEException )` [pure virtual]

Sets the message ID.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

**Parameters**

<i>id</i>	the ID of the message
-----------	-----------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if an internal error occurs.
--	--------------------------------

**6.515.3.32** `virtual void cms::Message::setCMSPriority ( int priority ) throw ( CMSEException )` [pure virtual]

Sets the Priority Value for this message.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

**Parameters**

<i>priority</i>	priority value for this message
-----------------	---------------------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 434), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 434), `activemq::commands::ActiveMQMessageTemplate< cms::Message >`

(p. 434), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 434), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 434), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 434).

**6.515.3.33** `virtual void cms::Message::setCMSRedelivered ( bool redelivered ) throw ( CMSEException )` [pure virtual]

Specifies whether this message is being redelivered.

This field is set at the time the message is delivered. This method can be used to change the value for a message that has been received.

#### Parameters

<i>redelivered</i>	boolean redelivered value
--------------------	---------------------------

#### Exceptions

<b>CMSEException</b> (p. 1190)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage`  
 > (p. 434), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage`  
 > (p. 434), `activemq::commands::ActiveMQMessageTemplate< cms::Message` >  
 (p. 434), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 434), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 434), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 434).

**6.515.3.34** `virtual void cms::Message::setCMSReplyTo ( const cms::Destination * destination ) throw ( CMSEException )` [pure virtual]

Sets the **Destination** (p. 1776) object to which a reply to this message should be sent.

The CMSReplyTo header field contains the destination where a reply to the current message should be sent. If it is null, no reply is expected. The destination may be either a **Queue** (p. 3238) object or a **Topic** (p. 3930) object.

Messages sent with a null CMSReplyTo value may be a notification of some event, or they may just be some data the sender thinks is of interest.

Messages with a CMSReplyTo value typically expect a response. A response is optional; it is up to the client to decide. These messages are called requests. A message sent in response to a request is called a reply.

In some cases a client may wish to match a request it sent earlier with a reply it has just received. The client can use the CMSCorrelationID header field for this purpose.

#### Parameters

<i>destination</i>	<b>Destination</b> (p. 1776) to which to send a response to this message
--------------------	--



**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 434), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 434), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 434), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 434), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 434), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 434).

**6.515.3.35** `virtual void cms::Message::setCMSTimestamp ( long long timeStamp ) throw ( CMSEException )` [pure virtual]

Sets the message timestamp.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

**Parameters**

<i>timeStamp</i>	integer time stamp value
------------------	--------------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 435), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 435), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 435), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 435), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 435), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 435).

**6.515.3.36** `virtual void cms::Message::setCMSType ( const std::string & type ) throw ( CMSEException )` [pure virtual]

Sets the message type.

Some CMS providers use a message repository that contains the definitions of messages sent by applications. The CMSType header field may reference a message's definition in the provider's repository.

The CMS API does not define a standard message definition repository, nor does it define a naming policy for the definitions it contains.

Some messaging systems require that a message type definition for each application

message be created and that each message specify its type. In order to work with such CMS providers, CMS clients should assign a value to CMSType, whether the application makes use of it or not. This ensures that the field is properly set for those providers that require it.

To ensure portability, CMS clients should use symbolic values for CMSType that can be configured at installation time to the values defined in the current provider's message repository. If string literals are used, they may not be valid type names for some CMS providers.

### Parameters

<i>type</i>	the message type
-------------	------------------

### See also

**getCMSType** (p. 2627)

### Exceptions

<b>CMSEException</b> (p. 1190)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 435), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 435), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 435), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 435), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 435), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 435).

**6.515.3.37** `virtual void cms::Message::setDoubleProperty ( const std::string & name, double value ) throw ( MessageNotWriteableException, CMSEException )`  
[pure virtual]

Sets a double property.

### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

### Exceptions

<b>CMSEException</b> (p. 1190)	- if the name is an empty string.
<b>MessageNotWriteableException</b> (p. 2808)	- if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 435), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 435), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 435), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 435), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 435), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 435).

**6.515.3.38** `virtual void cms::Message::setFloatProperty ( const std::string & name, float value ) throw ( MessageNotWriteableException, CMSException ) [pure virtual]`

Sets a float property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the name is an empty string.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 436), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 436), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 436), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 436), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 436), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 436).

**6.515.3.39** `virtual void cms::Message::setIntProperty ( const std::string & name, int value ) throw ( MessageNotWriteableException, CMSException ) [pure virtual]`

Sets a int property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the name is an empty string.
---	-----------------------------------

<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if properties are read-only
---	-------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 436), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 436), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 436), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 436), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 436), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 436).

**6.515.3.40** `virtual void cms::Message::setLongProperty ( const std::string & name, long long value ) throw ( MessageNotWriteableException, CMSEException )`  
[pure virtual]

Sets a long property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if the name is an empty string.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 436), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 436), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 436), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 436), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 436), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 436).

**6.515.3.41** `virtual void cms::Message::setShortProperty ( const std::string & name, short value ) throw ( MessageNotWriteableException, CMSEException )` [pure virtual]

Sets a short property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	- if the name is an empty string.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 437), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 437), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 437), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 437), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 437), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 437).

**6.515.3.42** `virtual void cms::Message::setStringProperty ( const std::string & name, const std::string & value ) throw ( MessageNotWriteableException, CMSException ) [pure virtual]`

Sets a string property.

**Parameters**

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	- if the name is an empty string.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 437), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 437), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 437), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 437), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 437), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 437).

The documentation for this class was generated from the following file:

- `src/main/cms/Message.h`

## 6.516 activemq::commands::MessageAck Class Reference

```
#include <src/main/activemq/commands/MessageAck.h>
```

Inheritance diagram for `activemq::commands::MessageAck`:

### Public Member Functions

- **MessageAck** ()
- virtual **~MessageAck** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **MessageAck \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual unsigned char **getAckType** () const
- virtual void **setAckType** (unsigned char ackType)
- virtual const **Pointer**< **MessageId** > & **getFirstMessageId** () const
- virtual **Pointer**< **MessageId** > & **getFirstMessageId** ()
- virtual void **setFirstMessageId** (const **Pointer**< **MessageId** > &firstMessageId)

- virtual const **Pointer**< **MessageId** > & **getLastMessageId** () const
- virtual **Pointer**< **MessageId** > & **getLastMessageId** ()
- virtual void **setLastMessageId** (const **Pointer**< **MessageId** > &lastMessageId)
- virtual int **getMessageCount** () const
- virtual void **setMessageCount** (int messageCount)
- virtual bool **isMessageAck** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_MESSAGEACK** = 22

### Protected Attributes

- **Pointer**< **ActiveMQDestination** > **destination**
- **Pointer**< **TransactionId** > **transactionId**
- **Pointer**< **ConsumerId** > **consumerId**
- unsigned char **ackType**
- **Pointer**< **MessageId** > **firstMessageId**
- **Pointer**< **MessageId** > **lastMessageId**
- int **messageCount**

## 6.516.1 Constructor & Destructor Documentation

6.516.1.1 **activemq::commands::MessageAck::MessageAck** ( )

6.516.1.2 **virtual activemq::commands::MessageAck::~~MessageAck** ( ) [virtual]

## 6.516.2 Member Function Documentation

6.516.2.1 **virtual MessageAck\* activemq::commands::MessageAck::cloneDataStructure** ( )  
const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

**6.516.2.2** `virtual void activemq::commands::MessageAck::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

**6.516.2.3** `virtual bool activemq::commands::MessageAck::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

**6.516.2.4** `virtual unsigned char activemq::commands::MessageAck::getAckType ( ) const [virtual]`

**6.516.2.5** `virtual const Pointer<ConsumerId>& activemq::commands::MessageAck::getConsumerId ( ) const [virtual]`

**6.516.2.6** `virtual Pointer<ConsumerId>& activemq::commands::MessageAck::getConsumerId ( ) [virtual]`

**6.516.2.7** `virtual unsigned char activemq::commands::MessageAck::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1713) type copy.

Implements `activemq::commands::DataStructure` (p. 1717).



- 6.516.2.8 **virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageAck::getDestination ( ) const** [virtual]
- 6.516.2.9 **virtual Pointer<ActiveMQDestination>& activemq::commands::MessageAck::getDestination ( )** [virtual]
- 6.516.2.10 **virtual const Pointer<MessageId>& activemq::commands::MessageAck::getFirstMessageId ( ) const** [virtual]
- 6.516.2.11 **virtual Pointer<MessageId>& activemq::commands::MessageAck::getFirstMessageId ( )** [virtual]
- 6.516.2.12 **virtual const Pointer<MessageId>& activemq::commands::MessageAck::getLastMessageId ( ) const** [virtual]
- 6.516.2.13 **virtual Pointer<MessageId>& activemq::commands::MessageAck::getLastMessageId ( )** [virtual]
- 6.516.2.14 **virtual int activemq::commands::MessageAck::getMessageCount ( ) const** [virtual]
- 6.516.2.15 **virtual const Pointer<TransactionId>& activemq::commands::MessageAck::getTransactionId ( ) const** [virtual]
- 6.516.2.16 **virtual Pointer<TransactionId>& activemq::commands::MessageAck::getTransactionId ( )** [virtual]
- 6.516.2.17 **virtual bool activemq::commands::MessageAck::isMessageAck ( ) const** [inline, virtual]

### Returns

an answer of true to the `isMessageAck()` (p. 2646) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 768).

- 6.516.2.18 `virtual void activemq::commands::MessageAck::setAckType ( unsigned char ackType ) [virtual]`
- 6.516.2.19 `virtual void activemq::commands::MessageAck::setConsumerId ( const Pointer< ConsumerId > & consumerId ) [virtual]`
- 6.516.2.20 `virtual void activemq::commands::MessageAck::setDestination ( const Pointer< ActiveMQDestination > & destination ) [virtual]`
- 6.516.2.21 `virtual void activemq::commands::MessageAck::setFirstMessageld ( const Pointer< MessageId > & firstMessageld ) [virtual]`
- 6.516.2.22 `virtual void activemq::commands::MessageAck::setLastMessageld ( const Pointer< MessageId > & lastMessageld ) [virtual]`
- 6.516.2.23 `virtual void activemq::commands::MessageAck::setMessageCount ( int messageCount ) [virtual]`
- 6.516.2.24 `virtual void activemq::commands::MessageAck::setTransactionId ( const Pointer< TransactionId > & transactionId ) [virtual]`
- 6.516.2.25 `virtual std::string activemq::commands::MessageAck::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

- 6.516.2.26 `virtual Pointer< Command > activemq::commands::MessageAck::visit ( activemq::state::CommandVisitor * visitor ) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

### 6.516.3 Field Documentation

- 6.516.3.1 `unsigned char activemq::commands::MessageAck::ackType`  
[protected]
- 6.516.3.2 `Pointer<ConsumerId> activemq::commands::MessageAck::consumerId`  
[protected]
- 6.516.3.3 `Pointer<ActiveMQDestination> activemq::commands::MessageAck::destination`  
[protected]
- 6.516.3.4 `Pointer<MessageId> activemq::commands::MessageAck::firstMessageId`  
[protected]
- 6.516.3.5 `const unsigned char activemq::commands::MessageAck::ID_-MESSAGEACK = 22` [static]
- 6.516.3.6 `Pointer<MessageId> activemq::commands::MessageAck::lastMessageId`  
[protected]
- 6.516.3.7 `int activemq::commands::MessageAck::messageCount` [protected]
- 6.516.3.8 `Pointer<TransactionId> activemq::commands::MessageAck::transactionId`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessageAck.h`

## 6.517 activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2648).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/MessageAckMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller`:

### Public Member Functions

- `MessageAckMarshaller ()`
- `virtual ~MessageAckMarshaller ()`
- `virtual commands::DataStructure * createObject () const`

*Creates a new instance of this marshalable type.*

- virtual unsigned char **getDataStructureType** () const

*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.517.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2648). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.517.2 Constructor & Destructor Documentation

6.517.2.1 `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::MessageAckMarshaller ( ) [inline]`

6.517.2.2 `virtual activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::~MessageAckMarshaller ( ) [inline, virtual]`

## 6.517.3 Member Function Documentation

6.517.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.517.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.517.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 801).

**6.517.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 802).

**6.517.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 803).

**6.517.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
`[virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 804).

**6.517.3.7** `virtual void activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` `[virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 806).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/MessageAckMarshaller.h`

## 6.518 `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller` Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2653).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarsh
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller`:

### Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )



*Write a object instance to data output stream.*

### 6.518.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2653). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.518.2 Constructor & Destructor Documentation

**6.518.2.1** `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::MessageAckMarshaller ( ) [inline]`

**6.518.2.2** `virtual activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::~~MessageAckMarshaller ( ) [inline, virtual]`

### 6.518.3 Member Function Documentation

**6.518.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.518.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.518.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 808).

```
6.518.3.4  virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 809).

```
6.518.3.5  virtual int activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 810).

**6.518.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 811).

**6.518.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 813).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h`

## 6.519 **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller** Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2657).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarsh
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller**:

**Public Member Functions**

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.519.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2657). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.519.2 Constructor & Destructor Documentation

- 6.519.2.1 **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::MessageAckMarshaller**  
( ) [inline]
- 6.519.2.2 **virtual activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::~~MessageAckMarshaller**  
( ) [inline, virtual]

### 6.519.3 Member Function Documentation

- 6.519.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

- 6.519.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

```
6.519.3.3 virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 772).

```
6.519.3.4 virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 774).

6.519.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 775).

6.519.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 776).

```
6.519.3.7 virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h

## 6.520 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2661).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarsh
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller**:

#### Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*



- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.520.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2661). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.520.2 Constructor & Destructor Documentation

6.520.2.1 **activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::MessageAckMarshaller**  
( ) [inline]

6.520.2.2 **virtual activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::~~MessageAckMarshaller**  
( ) [inline, virtual]

### 6.520.3 Member Function Documentation

6.520.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.520.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.520.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 780).

**6.520.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 781).

**6.520.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 782).

**6.520.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 783).

**6.520.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h`

## 6.521 **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller** Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2665).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarsh
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller**:

## Public Member Functions

- **MessageAckMarshaller ()**
- virtual **~MessageAckMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.521.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2665). NOTE!  
This file is auto generated - do not modify! if you need to make a change, please see  
the Java Classes in the activemq-openwire-generator module

## 6.521.2 Constructor & Destructor Documentation

6.521.2.1 `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::MessageAckMarshaller ( )` `[inline]`

6.521.2.2 `virtual activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::~~MessageAckMarshaller ( )` `[inline, virtual]`

## 6.521.3 Member Function Documentation

6.521.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::createObject ( )`  
`const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.521.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::getDataStructureType ( )` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.521.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut )` `throw ( decaf::io::IOException )` `[virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 787).

**6.521.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 788).

**6.521.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 789).

```
6.521.3.6 virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 790).

```
6.521.3.7 virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 792).

The documentation for this class was generated from the following file:



- src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h

## 6.522 activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2670).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller:

### Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.522.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2670). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.522.2 Constructor & Destructor Documentation

6.522.2.1 **activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::MessageAckMarshaller**  
( ) [inline]

6.522.2.2 **virtual activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::~~MessageAckMarshaller**  
( ) [inline, virtual]

### 6.522.3 Member Function Documentation

6.522.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.522.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.522.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 794).

**6.522.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 795).

**6.522.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 796).

```
6.522.3.6 virtual void activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 797).

```
6.522.3.7 virtual void activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h

**6.523 cms::MessageConsumer Class Reference**

A client uses a **MessageConsumer** (p. 2674) to received messages from a destination.

```
#include <src/main/cms/MessageConsumer.h>
```

Inheritance diagram for cms::MessageConsumer:

**Public Member Functions**

- virtual **~MessageConsumer** ()
- virtual **Message \* receive** ()=0 throw ( CMSEException )  
*Synchronously Receive a Message* (p. 2614).
- virtual **Message \* receive** (int millisecs)=0 throw ( CMSEException )  
*Synchronously Receive a Message* (p. 2614), time out after defined interval.
- virtual **Message \* receiveNoWait** ()=0 throw ( CMSEException )  
*Receive a Message* (p. 2614), does not wait if there isn't a new message to read, returns NULL if nothing read.
- virtual void **setMessageListener** (MessageListener \*listener)=0 throw ( CMSEException )  
*Sets the MessageListener* (p. 2779) that this class will send notifis on.
- virtual **MessageListener \* getMessageListener** () const =0 throw ( CMSEException )  
*Gets the MessageListener* (p. 2779) that this class will send mew Message (p. 2614) notification events to.
- virtual std::string **getMessageSelector** () const =0 throw ( cms::CMSEException )  
*Gets this message consumer's message selector expression.*

### 6.523.1 Detailed Description

A client uses a **MessageConsumer** (p. 2674) to received messages from a destination. A client may either synchronously receive a message consumer's messages or have the consumer asynchronously deliver them as they arrive.

For synchronous receipt, a client can request the next message from a message consumer using one of its `receive` methods. There are several variations of `receive` that allow a client to poll or wait for the next message.

For asynchronous delivery, a client can register a **MessageListener** (p. 2779) object with a message consumer. As messages arrive at the message consumer, it delivers them by calling the **MessageListener** (p. 2779)'s `onMessage` method.

When the `MessageConsumer`'s `close` method is called the method can block while an asynchronous message delivery is in progress or until a `receive` operation completes. A blocked consumer in a `receive` call will return a `Null` when the `close` method is called.

#### See also

**MessageListener** (p. 2779)

#### Since

1.0

### 6.523.2 Constructor & Destructor Documentation

6.523.2.1 `virtual cms::MessageConsumer::~~MessageConsumer ( ) [inline, virtual]`

### 6.523.3 Member Function Documentation

6.523.3.1 `virtual MessageListener* cms::MessageConsumer::getMessageListener ( ) const throw ( CMSEException ) [pure virtual]`

Gets the **MessageListener** (p. 2779) that this class will send new **Message** (p. 2614) notification events to.

#### Returns

The listener of messages received by this consumer

#### Exceptions

<b>CMSEException</b> (p. 1190)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedConsumer** (p. 1099), and **activemq::core::ActiveMQConsumer** (p. 307).

**6.523.3.2** `virtual std::string cms::MessageConsumer::getMessageSelector ( ) const throw ( cms::CMSEException ) [pure virtual]`

Gets this message consumer's message selector expression.

#### Returns

This Consumer's selector expression or "".

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- If an internal error occurs.
--	--------------------------------

Implemented in **activemq::cmsutil::CachedConsumer** (p. 1099), and **activemq::core::ActiveMQConsumer** (p. 307).

**6.523.3.3** `virtual Message* cms::MessageConsumer::receive ( int millisecs ) throw ( CMSEException ) [pure virtual]`

Synchronously Receive a **Message** (p. 2614), time out after defined interval.

Returns null if nothing read.

#### Returns

new message which the caller owns and must delete.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- If an internal error occurs.
--	--------------------------------

Implemented in **activemq::cmsutil::CachedConsumer** (p. 1100), and **activemq::core::ActiveMQConsumer** (p. 308).

**6.523.3.4** `virtual Message* cms::MessageConsumer::receive ( ) throw ( CMSEException ) [pure virtual]`

Synchronously Receive a **Message** (p. 2614).

#### Returns

new message which the caller owns and must delete.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- If an internal error occurs.
--	--------------------------------

Implemented in **activemq::cmsutil::CachedConsumer** (p. 1100), and **activemq::core::ActiveMQConsumer** (p. 308).

**6.523.3.5** `virtual Message* cms::MessageConsumer::receiveNoWait ( ) throw ( CMSEException ) [pure virtual]`

Receive a **Message** (p. 2614), does not wait if there isn't a new message to read, returns NULL if nothing read.

#### Returns

new message which the caller owns and must delete.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- If an internal error occurs.
--	--------------------------------

Implemented in **activemq::cmsutil::CachedConsumer** (p. 1100), and **activemq::core::ActiveMQConsumer** (p. 309).

**6.523.3.6** `virtual void cms::MessageConsumer::setMessageListener ( MessageListener * listener ) throw ( CMSEException ) [pure virtual]`

Sets the **MessageListener** (p. 2779) that this class will send notifs on.

#### Parameters

<i>listener</i>	The listener of messages received by this consumer.
-----------------	---

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- If an internal error occurs.
--	--------------------------------

Implemented in **activemq::cmsutil::CachedConsumer** (p. 1101), and **activemq::core::ActiveMQConsumer** (p. 309).

The documentation for this class was generated from the following file:

- `src/main/cms/MessageConsumer.h`

## 6.524 activemq::cmsutil::MessageCreator Class Reference

Creates the user-defined message to be sent by the **CmsTemplate** (p. 1201).

```
#include <src/main/activemq/cmsutil/MessageCreator.h>
```



## Public Member Functions

- virtual `~MessageCreator()`
- virtual `cms::Message * createMessage (cms::Session *session)=0` throw ( cms::CMSException )

*Creates a message from the given session.*

### 6.524.1 Detailed Description

Creates the user-defined message to be sent by the `CmsTemplate` (p. 1201).

### 6.524.2 Constructor & Destructor Documentation

6.524.2.1 virtual `activemq::cmsutil::MessageCreator::~MessageCreator ( )` [inline, virtual]

### 6.524.3 Member Function Documentation

6.524.3.1 virtual `cms::Message* activemq::cmsutil::MessageCreator::createMessage ( cms::Session * session )` throw ( cms::CMSException ) [pure virtual]

Creates a message from the given session.

#### Parameters

<i>session</i>	the CMS Session
----------------	-----------------

#### Exceptions

<i>cms::CMSException</i> (p. 1190)	if thrown by CMS API methods
---------------------------------------	------------------------------

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/MessageCreator.h`

## 6.525 activemq::commands::MessageDispatch Class Reference

```
#include <src/main/activemq/commands/MessageDispatch.h>
```

Inheritance diagram for `activemq::commands::MessageDispatch`:

## Public Member Functions

- **MessageDispatch** ()
- virtual **~MessageDispatch** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **MessageDispatch \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **Message** > & **getMessage** () const
- virtual **Pointer**< **Message** > & **getMessage** ()
- virtual void **setMessage** (const **Pointer**< **Message** > &message)
- virtual int **getRedeliveryCounter** () const
- virtual void **setRedeliveryCounter** (int redeliveryCounter)
- virtual bool **isMessageDispatch** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

## Static Public Attributes

- static const unsigned char **ID\_MESSAGEDISPATCH** = 21

### Protected Attributes

- **Pointer< ConsumerId > consumerId**
- **Pointer< ActiveMQDestination > destination**
- **Pointer< Message > message**
- **int redeliveryCounter**

### 6.525.1 Constructor & Destructor Documentation

6.525.1.1 **activemq::commands::MessageDispatch::MessageDispatch ( )**

6.525.1.2 **virtual activemq::commands::MessageDispatch::~~MessageDispatch ( )**  
[virtual]

### 6.525.2 Member Function Documentation

6.525.2.1 **virtual MessageDispatch\* activemq::commands::MessageDispatch::cloneDataStructure ( ) const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

6.525.2.2 **virtual void activemq::commands::MessageDispatch::copyDataStructure ( const DataStructure \* src )** [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

6.525.2.3 **virtual bool activemq::commands::MessageDispatch::equals ( const DataStructure \* value ) const** [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

**6.525.2.4** `virtual const Pointer<ConsumerId>& activemq::commands::MessageDispatch::getConsumerId ( ) const [virtual]`

**6.525.2.5** `virtual Pointer<ConsumerId>& activemq::commands::MessageDispatch::getConsumerId ( ) [virtual]`

**6.525.2.6** `virtual unsigned char activemq::commands::MessageDispatch::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns**

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

**6.525.2.7** `virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageDispatch::getDestination ( ) const [virtual]`

**6.525.2.8** `virtual Pointer<ActiveMQDestination>& activemq::commands::MessageDispatch::getDestination ( ) [virtual]`

**6.525.2.9** `virtual const Pointer<Message>& activemq::commands::MessageDispatch::getMessage ( ) const [virtual]`

**6.525.2.10** `virtual Pointer<Message>& activemq::commands::MessageDispatch::getMessage ( ) [virtual]`

**6.525.2.11** `virtual int activemq::commands::MessageDispatch::getRedeliveryCounter ( ) const [virtual]`

**6.525.2.12** `virtual bool activemq::commands::MessageDispatch::isMessageDispatch ( ) const [inline, virtual]`

**Returns**

an answer of true to the **isMessageDispatch()** (p. 2681) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 768).

- 6.525.2.13 `virtual void activemq::commands::MessageDispatch::setConsumerId ( const Pointer< ConsumerId > & consumerId ) [virtual]`
- 6.525.2.14 `virtual void activemq::commands::MessageDispatch::setDestination ( const Pointer< ActiveMQDestination > & destination ) [virtual]`
- 6.525.2.15 `virtual void activemq::commands::MessageDispatch::setMessage ( const Pointer< Message > & message ) [virtual]`
- 6.525.2.16 `virtual void activemq::commands::MessageDispatch::setRedeliveryCounter ( int redeliveryCounter ) [virtual]`
- 6.525.2.17 `virtual std::string activemq::commands::MessageDispatch::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

- 6.525.2.18 `virtual Pointer<Command> activemq::commands::MessageDispatch::visit ( activemq::state::CommandVisitor * visitor ) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

### 6.525.3 Field Documentation

- 6.525.3.1 **Pointer<ConsumerId> activemq::commands::MessageDispatch::consumerId**  
[protected]
- 6.525.3.2 **Pointer<ActiveMQDestination> activemq::commands::MessageDispatch::destination**  
[protected]
- 6.525.3.3 **const unsigned char activemq::commands::MessageDispatch::ID\_MESSAGEDISPATCH = 21** [static]
- 6.525.3.4 **Pointer<Message> activemq::commands::MessageDispatch::message**  
[protected]
- 6.525.3.5 **int activemq::commands::MessageDispatch::redeliveryCounter**  
[protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/MessageDispatch.h

## 6.526 activemq::core::MessageDispatchChannel Class Reference

```
#include <src/main/activemq/core/MessageDispatchChannel.h>
```

Inheritance diagram for activemq::core::MessageDispatchChannel:

### Public Member Functions

- **MessageDispatchChannel ()**
- **virtual ~MessageDispatchChannel ()**
- **void enqueue (const Pointer< MessageDispatch > &message)**  
*Add a Message to the Channel behind all pending message.*
- **void enqueueFirst (const Pointer< MessageDispatch > &message)**  
*Add a message to the front of the Channel.*
- **bool isEmpty () const**
- **bool isClosed () const**
- **bool isRunning () const**
- **Pointer< MessageDispatch > dequeue (long long timeout) throw ( exceptions::ActiveMQException )**  
*Used to get an enqueued message.*

- **Pointer< MessageDispatch > dequeueNoWait ()**

*Used to get an enqueued message if there is one queued right now.*

- **Pointer< MessageDispatch > peek () const**

*Peek in the Queue and return the first message in the Channel without removing it from the channel.*

- **void start ()**

*Starts dispatch of messages from the Channel.*

- **void stop ()**

*Stops dispatch of message from the Channel.*

- **void close ()**

*Close this channel no messages will be dispatched after this method is called.*

- **void clear ()**

*Clear the Channel, all pending messages are removed.*

- **int size () const**

- **std::vector< Pointer< MessageDispatch > > removeAll ()**

*Remove all messages that are currently in the Channel and return them as a list of Messages.*

- **virtual void lock () throw ( decaf::lang::exceptions::RuntimeException )**

*Locks the object.*

- **virtual bool tryLock () throw ( decaf::lang::exceptions::RuntimeException )**

*Attempts to Lock the object, if the lock is already held by another thread than this method returns false.*

- **virtual void unlock () throw ( decaf::lang::exceptions::RuntimeException )**

*Unlocks the object.*

- **virtual void wait () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )**

*Waits on a signal from this object, which is generated by a call to Notify.*

- **virtual void wait (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )**

*Waits on a signal from this object, which is generated by a call to Notify.*

- **virtual void wait (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )**

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals a waiter on this object that it can now wake up and continue.*

- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals the waiters on this object that it can now wake up and continue.*

### 6.526.1 Constructor & Destructor Documentation

6.526.1.1 **activemq::core::MessageDispatchChannel::MessageDispatchChannel ( )**

6.526.1.2 **virtual activemq::core::MessageDispatchChannel::~~MessageDispatchChannel ( )**  
[virtual]

### 6.526.2 Member Function Documentation

6.526.2.1 **void activemq::core::MessageDispatchChannel::clear ( )**

Clear the Channel, all pending messages are removed.

6.526.2.2 **void activemq::core::MessageDispatchChannel::close ( )**

Close this channel no messages will be dispatched after this method is called.

6.526.2.3 **Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::dequeue ( long long timeout ) throw ( exceptions::ActiveMQException )**

Used to get an enqueued message.

The amount of time this method blocks is based on the timeout value. - if timeout== -1 then it blocks until a message is received. - if timeout==0 then it tries to not block at all, it returns a message if it is available - if timeout>0 then it blocks up to timeout amount of time. Expired messages will be consumed by this method.

### Returns

null if we timeout or if the consumer is closed.

### Exceptions

<i>ActiveMQException</i>
--------------------------



**6.526.2.4** `Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::dequeueNoWait ( )`

Used to get an enqueued message if there is one queued right now.

If there is no waiting message than this method returns Null.

#### Returns

a message if there is one in the queue.

**6.526.2.5** `void activemq::core::MessageDispatchChannel::enqueue ( const Pointer<MessageDispatch> & message )`

Add a Message to the Channel behind all pending message.

#### Parameters

<i>message</i>	- The message to add to the Channel.
----------------	--------------------------------------

**6.526.2.6** `void activemq::core::MessageDispatchChannel::enqueueFirst ( const Pointer<MessageDispatch> & message )`

Add a message to the front of the Channel.

#### Parameters

<i>message</i>	- The Message to add to the front of the Channel.
----------------	---

**6.526.2.7** `bool activemq::core::MessageDispatchChannel::isClosed ( ) const` `[inline]`

#### Returns

has the Queue been closed.

**6.526.2.8** `bool activemq::core::MessageDispatchChannel::isEmpty ( ) const`

#### Returns

true if there are no messages in the Channel.

**6.526.2.9** `bool activemq::core::MessageDispatchChannel::isRunning ( ) const` `[inline]`

#### Returns

true if the Channel currently running and will dequeue message.

**6.526.2.10** `virtual void activemq::core::MessageDispatchChannel::lock ( ) throw ( decaf::lang::exceptions::RuntimeException )` [inline, virtual]

Locks the object.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3812).

**6.526.2.11** `virtual void activemq::core::MessageDispatchChannel::notify ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )` [inline, virtual]

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

#### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3813).

**6.526.2.12** `virtual void activemq::core::MessageDispatchChannel::notifyAll ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )` [inline, virtual]

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

#### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3814).

**6.526.2.13** `Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::peek ( ) const`

Peek in the Queue and return the first message in the Channel without removing it from the channel.

#### Returns

a message if there is one in the queue.

**6.526.2.14** `std::vector< Pointer<MessageDispatch> > activemq::core::MessageDispatchChannel::removeAll ( )`

Remove all messages that are currently in the Channel and return them as a list of Messages.

#### Returns

a list of Messages that was previously in the Channel.

**6.526.2.15** `int activemq::core::MessageDispatchChannel::size ( ) const`

#### Returns

the number of Messages currently in the Channel.

**6.526.2.16** `void activemq::core::MessageDispatchChannel::start ( )`

Starts dispatch of messages from the Channel.

**6.526.2.17** `void activemq::core::MessageDispatchChannel::stop ( )`

Stops dispatch of message from the Channel.

**6.526.2.18** `virtual bool activemq::core::MessageDispatchChannel::tryLock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

#### Returns

true if the lock was acquired, false if it is already held by another thread.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3815).

**6.526.2.19** `virtual void activemq::core::MessageDispatchChannel::unlock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Unlocks the object.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3816).

**6.526.2.20** `virtual void activemq::core::MessageDispatchChannel::wait ( long long millisecs ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3819).

**6.526.2.21** `virtual void activemq::core::MessageDispatchChannel::wait ( long long millisecs, int nanos ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the

6.527

**activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller****Class Reference****2693**

additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

**Parameters**

<i>milliseconds</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

**Exceptions**

<i>IllegalArgumentException</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3820).

```
6.526.2.22  virtual void activemq::core::MessageDispatchChannel::wait ( )
              throw ( decaf::lang::exceptions::RuntimeException,
                      decaf::lang::exceptions::IllegalMonitorStateException,
                      decaf::lang::exceptions::InterruptedException ) [inline,
                                virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

**Exceptions**

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3818).

The documentation for this class was generated from the following file:

- src/main/activemq/core/MessageDispatchChannel.h

## 6.527 **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller** **Class Reference**

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2690).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatch
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller`:

## Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.527.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2690).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.527

activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller

Class Reference

2695

## 6.527.2 Constructor & Destructor Documentation

6.527.2.1 `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::MessageDispatchMarshaller ( ) [inline]`

6.527.2.2 `virtual activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::~~MessageDispatchMarshaller ( ) [inline, virtual]`

## 6.527.3 Member Function Documentation

6.527.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.527.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.527.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 808).

**6.527.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 809).

**6.527.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



6.527

activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller

Class Reference

2697

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 810).

6.527.3.6 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::tightMarshal2 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** )  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 811).

6.527.3.7 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 813).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchMarshaller.h`

## 6.528 `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller` Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2695).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatch
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller`:

### Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual `~MessageDispatchMarshaller` ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

6.528

**activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller**

**Class Reference**

2699

---

*Write a object instance to data output stream.*

### 6.528.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2695).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.528.2 Constructor & Destructor Documentation

6.528.2.1 **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::MessageDispatchMarshaller**  
( ) [inline]

6.528.2.2 **virtual activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::~MessageDispatchMarshaller**  
( ) [inline, virtual]

### 6.528.3 Member Function Documentation

6.528.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.528.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.528.3.3 **virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 772).

**6.528.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 774).

**6.528.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

6.528

**activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller****Class Reference****2701****Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 775).

**6.528.3.6** virtual void **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::tightMarshal2**  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
 \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*,  
**utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** )  
 [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 776).

**6.528.3.7** virtual void **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::tightUnmarshal**  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream**  
 \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshaller.h`

## 6.529 **activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller** Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2699).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchch
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller**:

**Public Member Functions**

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

## 6.529

**activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller**

### Class Reference

2703

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.529.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2699).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.529.2 Constructor & Destructor Documentation

6.529.2.1 **activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::MessageDispatchMarshaller**  
( ) [inline]

6.529.2.2 **virtual activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::~~MessageDispatchMarshaller**  
( ) [inline, virtual]

### 6.529.3 Member Function Documentation

6.529.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.529.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

```
6.529.3.3  virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 794).

```
6.529.3.4  virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 795).



6.529

activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller

Class Reference

2705

6.529.3.5 virtual int activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::tightMarshal1 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 796).

6.529.3.6 virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::tightMarshal2 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 797).

```
6.529.3.7 virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchMarshaller.h

## 6.530 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2703).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller**:

#### Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

## 6.530

**activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller**

### Class Reference

2707

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.530.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2703).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.530.2 Constructor & Destructor Documentation

6.530.2.1 **activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::MessageDispatchMarshaller**  
( ) [inline]

6.530.2.2 **virtual activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::~~MessageDispatchMarshaller**  
( ) [inline, virtual]

### 6.530.3 Member Function Documentation

6.530.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.530.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.530.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 780).

**6.530.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

6.530

activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller

Class Reference

2709

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 781).

```
6.530.3.5  virtual int activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 782).

```
6.530.3.6  virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 783).

**6.530.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h`

## 6.531 **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller** Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2707).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatch
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller**:

- **MessageDispatchMarshaller ()**
- virtual **~MessageDispatchMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.531.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2707).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.531.2 Constructor & Destructor Documentation

6.531.2.1 `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::MessageDispatchMarshaller ( ) [inline]`

6.531.2.2 `virtual activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::~~MessageDispatchMarshaller ( ) [inline, virtual]`

### 6.531.3 Member Function Documentation

6.531.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.531.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.531.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



6.531

activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller

Class Reference

2713

Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 787).

**6.531.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 788).

**6.531.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 789).

```
6.531.3.6 virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 790).

```
6.531.3.7 virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 792).

The documentation for this class was generated from the following file:

6.532

**activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller**

**Class Reference**

2715

- `src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h`

## 6.532 **activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller** **Class Reference**

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2712).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller`:

### Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.532.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2712).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.532.2 Constructor & Destructor Documentation

6.532.2.1 **activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::MessageDispatchMarshaller**  
 ( ) [inline]

6.532.2.2 **virtual activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::~~MessageDispatchMarshaller**  
 ( ) [inline, virtual]

### 6.532.3 Member Function Documentation

6.532.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.532.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.532.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::looseMarshal**  
 ( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

6.532

**activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller**

**Class Reference**

2717

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 801).

**6.532.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 802).

**6.532.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 803).

```
6.532.3.6 virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 804).

```
6.532.3.7 virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## 6.533 activemq::commands::MessageDispatchNotification Class Reference 2719

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 806).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchMarshaller.h

## 6.533 activemq::commands::MessageDispatchNotification Class Reference

```
#include <src/main/activemq/commands/MessageDispatchNotification.h>
```

Inheritance diagram for **activemq::commands::MessageDispatchNotification**:

### Public Member Functions

- **MessageDispatchNotification** ()
- virtual **~MessageDispatchNotification** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **MessageDispatchNotification \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const

- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &**destination**)
- virtual long long **getDeliverySequenceId** () const
- virtual void **setDeliverySequenceId** (long long **deliverySequenceId**)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &**messageId**)
- virtual bool **isMessageDispatchNotification** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** \***visitor**)  
throw ( **exceptions::ActiveMQException** )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_MESSAGE\_DISPATCH\_NOTIFICATION** = 90

### Protected Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- long long **deliverySequenceId**
- **Pointer**< **MessageId** > **messageId**

## 6.533.1 Constructor & Destructor Documentation

6.533.1.1 **activemq::commands::MessageDispatchNotification::MessageDispatchNotification ( )**

6.533.1.2 **virtual activemq::commands::MessageDispatchNotification::~~MessageDispatchNotification ( ) [virtual]**

## 6.533.2 Member Function Documentation

6.533.2.1 **virtual MessageDispatchNotification\* activemq::commands::MessageDispatchNotification::cloneDataStructure ( ) const [virtual]**

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).



## 6.533 activemq::commands::MessageDispatchNotification Class Reference 2721

**6.533.2.2** `virtual void activemq::commands::MessageDispatchNotification::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

**6.533.2.3** `virtual bool activemq::commands::MessageDispatchNotification::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

**6.533.2.4** `virtual const Pointer<ConsumerId>& activemq::commands::MessageDispatchNotification::getConsumerId ( ) const [virtual]`

**6.533.2.5** `virtual Pointer<ConsumerId>& activemq::commands::MessageDispatchNotification::getConsumerId ( ) [virtual]`

**6.533.2.6** `virtual unsigned char activemq::commands::MessageDispatchNotification::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

### Returns

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

6.533.2.7 `virtual long long activemq::commands::MessageDispatchNotification::getDeliverySequenceld ( ) const [virtual]`

6.533.2.8 `virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageDispatchNotification::getDestination ( ) const [virtual]`

6.533.2.9 `virtual Pointer<ActiveMQDestination>& activemq::commands::MessageDispatchNotification::getDestination ( ) [virtual]`

6.533.2.10 `virtual Pointer<MessageId>& activemq::commands::MessageDispatchNotification::getMessageId ( ) [virtual]`

6.533.2.11 `virtual const Pointer<MessageId>& activemq::commands::MessageDispatchNotification::getMessageId ( ) const [virtual]`

6.533.2.12 `virtual bool activemq::commands::MessageDispatchNotification::isMessageDispatchNotification ( ) const [inline, virtual]`

### Returns

an answer of true to the `isMessageDispatchNotification()` (p. 2719) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 768).

6.533.2.13 `virtual void activemq::commands::MessageDispatchNotification::setConsumerId ( const Pointer< ConsumerId > & consumerId ) [virtual]`

6.533.2.14 `virtual void activemq::commands::MessageDispatchNotification::setDeliverySequenceld ( long long deliverySequenceld ) [virtual]`

6.533.2.15 `virtual void activemq::commands::MessageDispatchNotification::setDestination ( const Pointer< ActiveMQDestination > & destination ) [virtual]`

6.533.2.16 `virtual void activemq::commands::MessageDispatchNotification::setMessageId ( const Pointer< MessageId > & messageId ) [virtual]`

6.533.2.17 `virtual std::string activemq::commands::MessageDispatchNotification::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

**6.534 activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller**  
**Class Reference** 2723  
 Reimplemented from **activemq::commands::BaseCommand** (p. 770).

---

**6.533.2.18** `virtual Pointer<Command> activemq::commands::MessageDispatchNotification::visit (activemq::state::CommandVisitor * visitor ) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

#### 6.533.3 Field Documentation

**6.533.3.1** `Pointer<ConsumerId> activemq::commands::MessageDispatchNotification::consumerId [protected]`

**6.533.3.2** `long long activemq::commands::MessageDispatchNotification::deliverySequenceId [protected]`

**6.533.3.3** `Pointer<ActiveMQDestination> activemq::commands::MessageDispatchNotification::destination [protected]`

**6.533.3.4** `const unsigned char activemq::commands::MessageDispatchNotification::ID_ - MESSAGEDISPATCHNOTIFICATION = 90 [static]`

**6.533.3.5** `Pointer<MessageId> activemq::commands::MessageDispatchNotification::messageId [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessageDispatchNotification.h`

## 6.534 activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2720).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchNotificationMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller`:

## Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual **~MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.534.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2720). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.534.2.1 `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller ( ) [inline]`

6.534.2.2 `virtual activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller ( ) [inline, virtual]`

### 6.534.3 Member Function Documentation

6.534.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.534.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.534.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 801).

**6.534.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 802).

**6.534.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

### 6.534 ac-

**activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller**

#### Class Reference

2727

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller**  
(p. 803).

```
6.534.3.6 virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataOutputStream * dataOut,
 utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller**  
(p. 804).

```
6.534.3.7 virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
 dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
 * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller**  
(p. 806).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchNotificationMarshaller.h`

## 6.535 `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller` Class Reference

Marshaling code for Open Wire Format for `MessageDispatchNotificationMarshaller` (p. 2725).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchNotificationMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller`:

### Public Member Functions

- `MessageDispatchNotificationMarshaller ()`
- `virtual ~MessageDispatchNotificationMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaller.*
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`



---

*Write a object instance to data output stream.*

### 6.535.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2725). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.535.2 Constructor & Destructor Documentation

6.535.2.1 **activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller**  
**( )** [*inline*]

6.535.2.2 **virtual activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::~~MessageDispatchNotificationMarshaller**  
**( )** [*inline, virtual*]

### 6.535.3 Member Function Documentation

6.535.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::createObject**  
**( ) const** [*virtual*]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.535.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::getDataStructureType**  
**( ) const** [*virtual*]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.535.3.3 **virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException )** [*virtual*]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 808).

**6.535.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 809).

**6.535.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**6.535 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller**  
**Class Reference** **2731**  
**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 810).

**6.535.3.6** virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::tightMarshal2 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 811).

**6.535.3.7** virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 813).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**MessageDispatchNotificationMarshaller.h**

## 6.536 **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller** Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2729).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchNotificationMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller**:

**Public Member Functions**

- **MessageDispatchNotificationMarshaller** ()
- virtual **~MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.536.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2729). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.536.2 Constructor & Destructor Documentation

- 6.536.2.1 **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller**  
( ) [inline]
- 6.536.2.2 **virtual activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::~~MessageDispatchNotificationMarshaller**  
( ) [inline, virtual]

### 6.536.3 Member Function Documentation

- 6.536.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

- 6.536.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.536.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 772).

**6.536.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 774).

6.536 ac-

activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller

Class Reference

2735

```
6.536.3.5 virtual int activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 775).

```
6.536.3.6 virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 776).

6.536.3.7 virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchNotificationMarshaller.h

## 6.537 activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2733).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller**:

#### Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual **~MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*



- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.537.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2733). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.537.2 Constructor & Destructor Documentation

- 6.537.2.1 **activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller**  
( ) [inline]
- 6.537.2.2 **virtual activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::~~MessageDispatchNotificationMarshaller**  
( ) [inline, virtual]

### 6.537.3 Member Function Documentation

- 6.537.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.537.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.537.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 780).

**6.537.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

## 6.537 activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller Class Reference 2739

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 781).

**6.537.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 782).

**6.537.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 783).

**6.537.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller.h`

## 6.538 **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller** Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2737).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchNotificationMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller**:

- **MessageDispatchNotificationMarshaller ()**
- virtual **~MessageDispatchNotificationMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.538.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2737). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.538.2 Constructor & Destructor Documentation

6.538.2.1 **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller**  
( ) [inline]

6.538.2.2 **virtual activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::~~MessageDispatchNotificationMarshaller**  
( ) [inline, virtual]

## 6.538.3 Member Function Documentation

6.538.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.538.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.538.3.3 **virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::looseMarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## 6.538 activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller

### Class Reference 2743

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 787).

**6.538.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 788).

**6.538.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 789).

```
6.538.3.6 virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 790).

```
6.538.3.7 virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 792).

The documentation for this class was generated from the following file:



6.539 ac-

activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller

Class Reference

2745

- [src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchNotificationMarshaller.h](#)

## 6.539 activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller**  
(p. 2742).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchNotificationMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller`:

### Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual **~MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.539.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2742). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.539.2 Constructor & Destructor Documentation

6.539.2.1 **activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller**  
( ) [inline]

6.539.2.2 **virtual activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::~~MessageDispatchNotificationMarshaller**  
( ) [inline, virtual]

### 6.539.3 Member Function Documentation

6.539.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.539.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.539.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

## 6.539 activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller

### Class Reference 2747

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 794).

**6.539.3.4** virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::looseUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 795).

**6.539.3.5** virtual int activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::tightMarshal1 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 796).

```
6.539.3.6 virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 797).

```
6.539.3.7 virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchNotificationMarshaller.h

**6.540 cms::MessageEnumeration Class Reference**

Defines an object that enumerates a collection of Messages.

```
#include <src/main/cms/MessageEnumeration.h>
```

Inheritance diagram for cms::MessageEnumeration:

**Public Member Functions**

- virtual **~MessageEnumeration** ()
- virtual bool **hasMoreMessages** ()=0

*Returns true if there are more **Message** (p. 2614) in the Browser that can be retrieved via the *nextMessage* method.*

- virtual **cms::Message \* nextMessage** ()=0 throw ( cms::CMSEException )

*Returns the Next **Message** (p. 2614) in the **Queue** (p. 3238) if one is present, if no more **Message**'s are available then an Exception is thrown.*

**6.540.1 Detailed Description**

Defines an object that enumerates a collection of Messages. The client calls the hasMoreMessages method to determine if a **Message** (p. 2614) is available. If a **Message** (p. 2614) is available the client calls the nextMessage method to retrieve that **Message** (p. 2614), calling nextMessage when a **Message** (p. 2614) is not available results in an exception.

**Since**

2.1

## 6.540.2 Constructor & Destructor Documentation

6.540.2.1 `virtual cms::MessageEnumeration::~~MessageEnumeration ( ) [inline, virtual]`

## 6.540.3 Member Function Documentation

6.540.3.1 `virtual bool cms::MessageEnumeration::hasMoreMessages ( ) [pure virtual]`

Returns true if there are more **Message** (p. 2614) in the Browser that can be retrieved via the `nextMessage` method.

If this method returns false and the `nextMessage` method is called then an Exception will be thrown.

### Returns

true if more **Message**'s are available in the Browser.

Implemented in `activemq::core::ActiveMQQueueBrowser` (p. 486).

6.540.3.2 `virtual cms::Message* cms::MessageEnumeration::nextMessage ( ) throw ( cms::CMSException ) [pure virtual]`

Returns the Next **Message** (p. 2614) in the **Queue** (p. 3238) if one is present, if no more **Message**'s are available then an Exception is thrown.

If a **Message** (p. 2614) object pointer is returned then that object becomes the property of the caller and must be deleted by the caller when finished.

### Returns

The next **Message** (p. 2614) in the **Queue** (p. 3238).

### Exceptions

<i><b>CMSException</b></i> (p. 1190)	if no more <b>Message</b> 's currently in the <b>Queue</b> (p. 3238).
---	---

Implemented in `activemq::core::ActiveMQQueueBrowser` (p. 486).

The documentation for this class was generated from the following file:

- `src/main/cms/MessageEnumeration.h`

## 6.541 cms::MessageEOFException Class Reference

This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 3760) or **BytesMessage** (p. 1079) is being read.

```
#include <src/main/cms/MessageEOFException.h>
```

Inheritance diagram for cms::MessageEOFException:

### Public Member Functions

- **MessageEOFException** () throw ()
- **MessageEOFException** (const **MessageEOFException** &ex) throw ()
- **MessageEOFException** (const std::string &message, const std::exception \*cause) throw ()
- **MessageEOFException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**MessageEOFException** () throw ()

#### 6.541.1 Detailed Description

This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 3760) or **BytesMessage** (p. 1079) is being read.

Since

1.3

#### 6.541.2 Constructor & Destructor Documentation

6.541.2.1 cms::MessageEOFException::MessageEOFException ( ) throw ()

6.541.2.2 cms::MessageEOFException::MessageEOFException ( const **MessageEOFException** & ex ) throw ()

6.541.2.3 cms::MessageEOFException::MessageEOFException ( const std::string & message, const std::exception \* cause ) throw ()

6.541.2.4 cms::MessageEOFException::MessageEOFException ( const std::string & message, const std::exception \* cause, const std::vector< std::pair< std::string, int > > & stackTrace ) throw ()

6.541.2.5 virtual cms::MessageEOFException::~MessageEOFException ( ) throw ()  
[virtual]

The documentation for this class was generated from the following file:

- src/main/cms/MessageEOFException.h

## 6.542 cms::MessageFormatException Class Reference

This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.

```
#include <src/main/cms/MessageFormatException.h>
```

Inheritance diagram for cms::MessageFormatException:

### Public Member Functions

- **MessageFormatException** () throw ()
- **MessageFormatException** (const **MessageFormatException** &ex) throw ()
- **MessageFormatException** (const std::string &message, const std::exception \*cause) throw ()
- **MessageFormatException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**MessageFormatException** () throw ()

### 6.542.1 Detailed Description

This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type. It must also be thrown when equivalent type errors are made with message property values. For example, this exception must be thrown if **StreamMessage.readShort** (p. 3769) is used to read a boolean value.

### Since

1.3



### 6.542.2 Constructor & Destructor Documentation

6.542.2.1 `cms::MessageFormatException::MessageFormatException ( ) throw ()`

6.542.2.2 `cms::MessageFormatException::MessageFormatException ( const MessageFormatException & ex ) throw ()`

6.542.2.3 `cms::MessageFormatException::MessageFormatException ( const std::string & message, const std::exception * cause ) throw ()`

6.542.2.4 `cms::MessageFormatException::MessageFormatException ( const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace ) throw ()`

6.542.2.5 `virtual cms::MessageFormatException::~MessageFormatException ( ) throw ()`  
[virtual]

The documentation for this class was generated from the following file:

- `src/main/cms/MessageFormatException.h`

## 6.543 activemq::commands::MessageId Class Reference

```
#include <src/main/activemq/commands/MessageId.h>
```

Inheritance diagram for `activemq::commands::MessageId`:

### Public Types

- `typedef decaf::lang::PointerComparator< MessageId > COMPARATOR`

### Public Member Functions

- `MessageId ()`
- `MessageId (const MessageId &other)`
- `MessageId (const std::string &messageKey)`
- `MessageId (const Pointer< ProducerInfo > &producerInfo, long long producerSequenceId)`
- `MessageId (const Pointer< ProducerId > &producerId, long long producerSequenceId)`
- `MessageId (const std::string &producerId, long long producerSequenceId)`
- `virtual ~MessageId ()`
- `virtual unsigned char getDataStructureType () const`

*Get the unique identifier that this object and its own Marshaler share.*

- virtual **MessageId \* cloneDataStructure ()** const

*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*

- virtual void **copyDataStructure** (const **DataStructure \*src**)

*Copy the contents of the passed object into this object's members, overwriting any existing data.*

- virtual std::string **toString ()** const

*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure \*value**) const

*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*

- void **setValue** (const std::string &key)
- void **setTextView** (const std::string &key)
- virtual const **Pointer< ProducerId > & getProducerId ()** const
- virtual **Pointer< ProducerId > & getProducerId ()**
- virtual void **setProducerId** (const **Pointer< ProducerId > &producerId**)
- virtual long long **getProducerSequenceId ()** const
- virtual void **setProducerSequenceId** (long long **producerSequenceId**)
- virtual long long **getBrokerSequenceId ()** const
- virtual void **setBrokerSequenceId** (long long **brokerSequenceId**)
- virtual int **compareTo** (const **MessageId &value**) const
- virtual bool **equals** (const **MessageId &value**) const
- virtual bool **operator==** (const **MessageId &value**) const
- virtual bool **operator<** (const **MessageId &value**) const
- **MessageId & operator=** (const **MessageId &other**)

### Static Public Attributes

- static const unsigned char **ID\_MESSAGEID = 110**

### Protected Attributes

- **Pointer< ProducerId > producerId**
- long long **producerSequenceId**
- long long **brokerSequenceId**

### 6.543.1 Member Typedef Documentation

6.543.1.1 `typedef decaf::lang::PointerComparator<MessageId>  
activemq::commands::MessageId::COMPARATOR`

### 6.543.2 Constructor & Destructor Documentation

6.543.2.1 `activemq::commands::MessageId::MessageId ( )`

6.543.2.2 `activemq::commands::MessageId::MessageId ( const MessageId & other )`

6.543.2.3 `activemq::commands::MessageId::MessageId ( const std::string & messageKey )`

6.543.2.4 `activemq::commands::MessageId::MessageId ( const Pointer< ProducerInfo >  
& producerInfo, long long producerSequenceId )`

6.543.2.5 `activemq::commands::MessageId::MessageId ( const Pointer< ProducerId > &  
producerId, long long producerSequenceId )`

6.543.2.6 `activemq::commands::MessageId::MessageId ( const std::string & producerId, long  
long producerSequenceId )`

6.543.2.7 `virtual activemq::commands::MessageId::~~MessageId ( )` [virtual]

### 6.543.3 Member Function Documentation

6.543.3.1 `virtual MessageId* activemq::commands::MessageId::cloneDataStructure ( ) const`  
[virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1714).

6.543.3.2 `virtual int activemq::commands::MessageId::compareTo ( const MessageId & value  
) const` [virtual]

6.543.3.3 `virtual void activemq::commands::MessageId::copyDataStructure ( const  
DataStructure * src )` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1715).

**6.543.3.4** `virtual bool activemq::commands::MessageId::equals ( const DataStructure *  
value ) const [virtual]`

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1716).

**6.543.3.5** `virtual bool activemq::commands::MessageId::equals ( const MessageId & value )  
const [virtual]`

**6.543.3.6** `virtual long long activemq::commands::MessageId::getBrokerSequenceId ( ) const  
[virtual]`

**6.543.3.7** `virtual unsigned char activemq::commands::MessageId::getDataStructureType ( )  
const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

- 6.543.3.8 `virtual const Pointer<ProducerId>& activemq::commands::MessageId::getProducerId ( ) const`  
[virtual]
- 6.543.3.9 `virtual Pointer<ProducerId>& activemq::commands::MessageId::getProducerId ( )` [virtual]
- 6.543.3.10 `virtual long long activemq::commands::MessageId::getProducerSequenceId ( ) const` [virtual]
- 6.543.3.11 `virtual bool activemq::commands::MessageId::operator< ( const MessageId & value ) const` [virtual]
- 6.543.3.12 `MessageId& activemq::commands::MessageId::operator= ( const MessageId & other )`
- 6.543.3.13 `virtual bool activemq::commands::MessageId::operator==( const MessageId & value ) const` [virtual]
- 6.543.3.14 `virtual void activemq::commands::MessageId::setBrokerSequenceId ( long long brokerSequenceId )` [virtual]
- 6.543.3.15 `virtual void activemq::commands::MessageId::setProducerId ( const Pointer<ProducerId> & producerId )` [virtual]
- 6.543.3.16 `virtual void activemq::commands::MessageId::setProducerSequenceId ( long long producerSequenceId )` [virtual]
- 6.543.3.17 `void activemq::commands::MessageId::setTextView ( const std::string & key )`
- 6.543.3.18 `void activemq::commands::MessageId::setValue ( const std::string & key )`
- 6.543.3.19 `virtual std::string activemq::commands::MessageId::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 841).

### 6.543.4 Field Documentation

- 6.543.4.1 `long long activemq::commands::MessageId::brokerSequenceId`  
[protected]
- 6.543.4.2 `const unsigned char activemq::commands::MessageId::ID_MESSAGEID`  
`= 110` [static]
- 6.543.4.3 `Pointer<ProducerId> activemq::commands::MessageId::producerId`  
[protected]
- 6.543.4.4 `long long activemq::commands::MessageId::producerSequenceId`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessageId.h`

## 6.544 `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller` Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2755).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarsha
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller`:

### Public Member Functions

- **MessageIdMarshaller** ()
- virtual `~MessageIdMarshaller` ()
- virtual `commands::DataStructure * createObject` () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataInputStream *dataIn`, `utils::BooleanStream *bs`)  
`throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `utils::BooleanStream *bs`) `throw ( decaf::io::IOException )`

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.544.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2755). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.544.2 Constructor & Destructor Documentation

6.544.2.1 **activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::MessageIdMarshaller**  
( ) [inline]

6.544.2.2 **virtual activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::~~MessageIdMarshaller**  
( ) [inline, virtual]

### 6.544.3 Member Function Documentation

6.544.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.544.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.544.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1675).

**6.544.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--



## 6.544 activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller

### Class Reference 2761

---

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

```
6.544.3.5 virtual int activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.544.3.6 virtual void activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.544.3.7 virtual void activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**MessageIdMarshaller.h**

## 6.545 activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2759).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarsha
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller**:

#### Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.545.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2759). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.545.2 Constructor & Destructor Documentation

6.545.2.1 **activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::MessageIdMarshaller**  
( ) [inline]

6.545.2.2 **virtual activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::~~MessageIdMarshaller**  
( ) [inline, virtual]

### 6.545.3 Member Function Documentation

6.545.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::createObject** ( )  
const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.545.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.545.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.545.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.545.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.545.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.545.3.7 virtual void activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
  * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h

## 6.546 activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2763).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarsha
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller**:

#### Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.546.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2763). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.546.2 Constructor & Destructor Documentation

6.546.2.1 **activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::MessageIdMarshaller**  
( ) [inline]

6.546.2.2 **virtual activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::~~MessageIdMarshaller**  
( ) [inline, virtual]

### 6.546.3 Member Function Documentation

6.546.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.546.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.546.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.546.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source



## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.546.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.546.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.546.3.7 virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
  * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarshaller.h

## 6.547 activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2767).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarsha
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller**:

#### Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.547.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2767). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.547.2 Constructor & Destructor Documentation

6.547.2.1 **activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::MessageIdMarshaller**  
( ) [inline]

6.547.2.2 **virtual activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::~~MessageIdMarshaller**  
( ) [inline, virtual]

### 6.547.3 Member Function Documentation

6.547.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.547.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.547.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.547.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.547.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.547.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.547.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h`

## 6.548 activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2771).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/MessageIdMarsha
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller**:

#### Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.548.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2771). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.548.2 Constructor & Destructor Documentation

6.548.2.1 **activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::MessageIdMarshaller**  
( ) [inline]

6.548.2.2 **virtual activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::~~MessageIdMarshaller**  
( ) [inline, virtual]

### 6.548.3 Member Function Documentation

6.548.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.548.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.548.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.548.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source



## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.548.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.548.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.548.3.7** `virtual void activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/MessageIdMarshaller.h`

## 6.549 activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2775).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarsha
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller**:

#### Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.549.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2775). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.549.2 Constructor & Destructor Documentation

6.549.2.1 **activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::MessageIdMarshaller**  
( ) [inline]

6.549.2.2 **virtual activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::~~MessageIdMarshaller**  
( ) [inline, virtual]

### 6.549.3 Member Function Documentation

6.549.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.549.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.549.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.549.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.549.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.549.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.549.3.7 virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
  * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**MessageIdMarshaller.h**

## 6.550 cms::MessageListener Class Reference

A **MessageListener** (p. 2779) object is used to receive asynchronously delivered messages.

```
#include <src/main/cms/MessageListener.h>
```

#### Public Member Functions

- virtual **~MessageListener** ()
- virtual void **onMessage** (const **Message** \*message)=0

*Called asynchronously when a new message is received, the message reference can be to any of the **Message** (p. 2614) types.*

#### 6.550.1 Detailed Description

A **MessageListener** (p. 2779) object is used to receive asynchronously delivered messages.

## 6.551 activemq::wireformat::openwire::marshal::v5::MessageMarshaller Class Reference 2783

---

Since

1.0

### 6.550.2 Constructor & Destructor Documentation

6.550.2.1 `virtual cms::MessageListener::~~MessageListener ( ) [inline, virtual]`

### 6.550.3 Member Function Documentation

6.550.3.1 `virtual void cms::MessageListener::onMessage ( const Message * message ) [pure virtual]`

Called asynchronously when a new message is received, the message reference can be to any of the **Message** (p. 2614) types.

a dynamic cast is used to find out what type of message this is. The lifetime of this object is only guaranteed to be for life of the onMessage function after this call-back returns the message may no longer exists. Users should copy the data or clone the message if they wish to retain information that was contained in this **Message** (p. 2614).

It is considered a programming error for this method to throw an exception.

#### Parameters

<i>message</i>	<b>Message</b> (p. 2614) object {const} pointer recipient does not own.
----------------	---

The documentation for this class was generated from the following file:

- src/main/cms/MessageListener.h

## 6.551 activemq::wireformat::openwire::marshal::v5::MessageMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2780).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::MessageMarshaller:

#### Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.551.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2780). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.551.2 Constructor & Destructor Documentation

6.551.2.1 **activemq::wireformat::openwire::marshal::v5::MessageMarshaller::MessageMarshaller**  
( ) [inline]

6.551.2.2 **virtual activemq::wireformat::openwire::marshal::v5::MessageMarshaller::~~MessageMarshaller**  
( ) [inline, virtual]

### 6.551.3 Member Function Documentation

6.551.3.1 **virtual void activemq::wireformat::openwire::marshal::v5::MessageMarshaller::looseMarshal**  
( **OpenWireFormat** \* wireFormat, **commands::DataStructure** \* dataStructure, **decaf::io::DataOutputStream** \* dataOut ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.



#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 794).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 208), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 249), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 379), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 407), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 455), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 567), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 686).

**6.551.3.2** `virtual void activemq::wireformat::openwire::marshal::v5::MessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 795).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 209), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 250), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 379), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 408), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 455), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 568), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller**

(p. 686).

```
6.551.3.3 virtual int activemq::wireformat::openwire::marshal::v5::MessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 796).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 209), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 250), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 380), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 408), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 456), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 568), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 687).

```
6.551.3.4 virtual void activemq::wireformat::openwire::marshal::v5::MessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 797).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 210), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 251), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 380), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 409), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 456), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 569), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 687).

**6.551.3.5** virtual void activemq::wireformat::openwire::marshal::v5::MessageMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 210), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 251), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 381), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 409), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 457), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 569), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 688).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h

## 6.552 activemq::wireformat::openwire::marshal::v3::MessageMarshaller

### Class Reference

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2785).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::MessageMarshaller:

#### Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

#### 6.552.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2785). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.552.2 Constructor & Destructor Documentation

6.552.2.1 `activemq::wireformat::openwire::marshal::v3::MessageMarshaller::MessageMarshaller ( ) [inline]`

6.552.2.2 `virtual activemq::wireformat::openwire::marshal::v3::MessageMarshaller::~MessageMarshaller ( ) [inline, virtual]`

## 6.552.3 Member Function Documentation

6.552.3.1 `virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 772).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 191), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 236), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 366), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 394), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 442), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 555), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 673).

6.552.3.2 `virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 774).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 192), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 237), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 367), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 395), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 443), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 555), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 674).

```
6.552.3.3  virtual int activemq::wireformat::openwire::marshal::v3::MessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 775).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 192), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 237), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 367), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 395), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller**

## 6.552 activemq::wireformat::openwire::marshal::v3::MessageMarshaller Class Reference 2791

(p. 443), [activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller](#) (p. 556), and [activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller](#) (p. 674).

```
6.552.3.4 virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataOutputStream * dataOut,
 utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from [activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller](#) (p. 776).

Reimplemented in [activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller](#) (p. 193), [activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller](#) (p. 238), [activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller](#) (p. 368), [activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller](#) (p. 396), [activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller](#) (p. 444), [activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller](#) (p. 556), and [activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller](#) (p. 675).

```
6.552.3.5 virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
 dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
 * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 777).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 193), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 238), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 368), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 396), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 444), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 557), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 675).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h`

## 6.553 `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` Class Reference

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2789).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::MessageMarshaller`:

**Public Member Functions**

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )



*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.553.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2789). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.553.2 Constructor & Destructor Documentation

6.553.2.1 **activemq::wireformat::openwire::marshal::v2::MessageMarshaller::MessageMarshaller**  
( ) [inline]

6.553.2.2 **virtual activemq::wireformat::openwire::marshal::v2::MessageMarshaller::~MessageMarshaller**  
( ) [inline, virtual]

### 6.553.3 Member Function Documentation

6.553.3.1 **virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::looseMarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 808).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 204), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 258), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 383), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 411), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 459), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 572), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 694).

**6.553.3.2** `virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 809).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 204), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 258), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 384), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 412), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 460), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 572), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 695).

**6.553.3.3** `virtual int activemq::wireformat::openwire::marshal::v2::MessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

## 6.553 activemq::wireformat::openwire::marshal::v2::MessageMarshaller Class Reference 2795

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 810).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 205), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 259), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 384), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 412), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 460), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 573), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 695).

```
6.553.3.4 virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 811).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 205), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller**

(p. 259), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 385), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 413), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 461), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 573), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 696).

**6.553.3.5** `virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 813).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 206), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 260), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 385), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 413), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 461), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 574), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 696).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h`

## 6.554 `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `MessageMarshaller` (p. 2793).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::MessageMarshaller:

## Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.554.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2793). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.554.2 Constructor & Destructor Documentation

6.554.2.1 `activemq::wireformat::openwire::marshal::v4::MessageMarshaller::MessageMarshaller ( ) [inline]`

6.554.2.2 `virtual activemq::wireformat::openwire::marshal::v4::MessageMarshaller::~~MessageMarshaller ( ) [inline, virtual]`

### 6.554.3 Member Function Documentation

6.554.3.1 `virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 780).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 200), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 245), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 375), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 403), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 451), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 563), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 677).

6.554.3.2 `virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

## 6.554 activemq::wireformat::openwire::marshal::v4::MessageMarshaller Class Reference 2799

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 781).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller** (p. 200), **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller** (p. 245), **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller** (p. 375), **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller** (p. 403), **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller** (p. 451), **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller** (p. 564), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller** (p. 678).

```
6.554.3.3  virtual int activemq::wireformat::openwire::marshal::v4::MessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 782).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller** (p. 201), **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller** (p. 246), **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller** (p. 376), **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller** (p. 404), **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller**

(p. 452), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 564), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 678).

**6.554.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 783).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 201), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 246), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 376), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 404), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 452), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 565), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 679).

**6.554.3.5** `virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.



## 6.555 activemq::wireformat::openwire::marshal::v1::MessageMarshaller Class Reference 2801

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 784).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller** (p. 202), **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller** (p. 247), **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller** (p. 377), **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller** (p. 405), **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller** (p. 453), **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller** (p. 565), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller** (p. 679).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h

## 6.555 activemq::wireformat::openwire::marshal::v1::MessageMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2798).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::MessageMarshaller**:

### Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.555.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2798). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.555.2 Constructor & Destructor Documentation

6.555.2.1 **activemq::wireformat::openwire::marshal::v1::MessageMarshaller::MessageMarshaller**  
( ) [inline]

6.555.2.2 **virtual activemq::wireformat::openwire::marshal::v1::MessageMarshaller::~~MessageMarshaller**  
( ) [inline, virtual]

### 6.555.3 Member Function Documentation

6.555.3.1 **virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::looseMarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.555 activemq::wireformat::openwire::marshal::v1::MessageMarshaller Class Reference 2803

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 787).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 195), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 241), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 370), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 399), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 446), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 559), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 682).

```
6.555.3.2  virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataInputStream * dataIn ) throw (
 decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 788).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 196), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 241), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 371), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 399), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 447), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 559), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 682).

```
6.555.3.3  virtual int activemq::wireformat::openwire::marshal::v1::MessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
 dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
 [virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 789).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 196), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 242), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 371), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 400), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 447), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 560), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 683).

```
6.555.3.4  virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 790).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 197), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller`

## 6.556 activemq::wireformat::openwire::marshal::v6::MessageMarshaller Class Reference

2805

(p. 242), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 372), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 400), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 448), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 560), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 683).

**6.555.3.5** `virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 792).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 197), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 243), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 372), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 401), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 448), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 561), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 684).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h`

## 6.556 activemq::wireformat::openwire::marshal::v6::MessageMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2802).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::MessageMarshaller`:

## Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.556.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2802). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.556.2 Constructor & Destructor Documentation

6.556.2.1 `activemq::wireformat::openwire::marshal::v6::MessageMarshaller::MessageMarshaller ( ) [inline]`

6.556.2.2 `virtual activemq::wireformat::openwire::marshal::v6::MessageMarshaller::~MessageMarshaller ( ) [inline, virtual]`

## 6.556.3 Member Function Documentation

6.556.3.1 `virtual void activemq::wireformat::openwire::marshal::v6::MessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 801).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 212), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 253), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 387), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 416), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 463), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 576), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 690).

6.556.3.2 `virtual void activemq::wireformat::openwire::marshal::v6::MessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 802).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller** (p. 213), **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller** (p. 254), **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller** (p. 388), **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller** (p. 416), **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller** (p. 464), **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller** (p. 576), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller** (p. 691).

```
6.556.3.3  virtual int activemq::wireformat::openwire::marshal::v6::MessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 803).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller** (p. 213), **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller** (p. 254), **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller** (p. 388), **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller** (p. 417), **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller**



## 6.556 activemq::wireformat::openwire::marshal::v6::MessageMarshaller Class Reference 2809

(p. 464), [activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller](#) (p. 577), and [activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller](#) (p. 691).

```
6.556.3.4 virtual void activemq::wireformat::openwire::marshal::v6::MessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataOutputStream * dataOut,
 utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from [activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller](#) (p. 804).

Reimplemented in [activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller](#) (p. 214), [activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller](#) (p. 255), [activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller](#) (p. 389), [activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller](#) (p. 417), [activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller](#) (p. 465), [activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller](#) (p. 577), and [activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller](#) (p. 692).

```
6.556.3.5 virtual void activemq::wireformat::openwire::marshal::v6::MessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
 dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
 * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 806).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 214), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 255), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 389), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 418), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 465), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 578), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 692).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h`

**6.557 cms::MessageNotReadableException Class Reference**

This exception must be thrown when a CMS client attempts to read a write-only message.

```
#include <src/main/cms/MessageNotReadableException.h>
```

Inheritance diagram for `cms::MessageNotReadableException`:

**Public Member Functions**

- **MessageNotReadableException** () throw ()
- **MessageNotReadableException** (const **MessageNotReadableException** &ex) throw ()
- **MessageNotReadableException** (const std::string &message, const std::exception \*cause) throw ()
- **MessageNotReadableException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual **~MessageNotReadableException** () throw ()

**6.557.1 Detailed Description**

This exception must be thrown when a CMS client attempts to read a write-only message.

**Since**

1.3

## 6.557.2 Constructor & Destructor Documentation

- 6.557.2.1 `cms::MessageNotReadableException::MessageNotReadableException ( ) throw ()`
- 6.557.2.2 `cms::MessageNotReadableException::MessageNotReadableException ( const MessageNotReadableException & ex ) throw ()`
- 6.557.2.3 `cms::MessageNotReadableException::MessageNotReadableException ( const std::string & message, const std::exception * cause ) throw ()`
- 6.557.2.4 `cms::MessageNotReadableException::MessageNotReadableException ( const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace ) throw ()`
- 6.557.2.5 `virtual cms::MessageNotReadableException::~MessageNotReadableException ( ) throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/MessageNotReadableException.h`

## 6.558 cms::MessageNotWriteableException Class Reference

This exception must be thrown when a CMS client attempts to write to a read-only message.

```
#include <src/main/cms/MessageNotWriteableException.h>
```

Inheritance diagram for cms::MessageNotWriteableException:

### Public Member Functions

- `MessageNotWriteableException () throw ()`
- `MessageNotWriteableException (const MessageNotWriteableException &ex) throw ()`
- `MessageNotWriteableException (const std::string &message, const std::exception *cause) throw ()`
- `MessageNotWriteableException (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()`
- `virtual ~MessageNotWriteableException () throw ()`

### 6.558.1 Detailed Description

This exception must be thrown when a CMS client attempts to write to a read-only message.

Since

1.3

## 6.558.2 Constructor & Destructor Documentation

6.558.2.1 `cms::MessageNotWriteableException::MessageNotWriteableException ( ) throw ()`

6.558.2.2 `cms::MessageNotWriteableException::MessageNotWriteableException ( const MessageNotWriteableException & ex ) throw ()`

6.558.2.3 `cms::MessageNotWriteableException::MessageNotWriteableException ( const std::string & message, const std::exception * cause ) throw ()`

6.558.2.4 `cms::MessageNotWriteableException::MessageNotWriteableException ( const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace ) throw ()`

6.558.2.5 `virtual cms::MessageNotWriteableException::~~MessageNotWriteableException ( ) throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/MessageNotWriteableException.h`

## 6.559 cms::MessageProducer Class Reference

A client uses a **MessageProducer** (p. 2809) object to send messages to a **Destination** (p. 1776).

```
#include <src/main/cms/MessageProducer.h>
```

Inheritance diagram for `cms::MessageProducer`:

### Public Member Functions

- `virtual ~MessageProducer ()`
- `virtual void send (Message *message)=0 throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )`  
*Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.*
- `virtual void send (Message *message, int deliveryMode, int priority, long long timeToLive)=0 throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )`

*Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.*

- virtual void **send** (const **Destination** \*destination, **Message** \*message)=0 throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )

*Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.*

- virtual void **send** (const **Destination** \*destination, **Message** \*message, int deliveryMode, int priority, long long timeToLive)=0 throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )

*Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.*

- virtual void **setDeliveryMode** (int mode)=0 throw ( CMSException )

*Sets the delivery mode for this Producer.*

- virtual int **getDeliveryMode** () const =0 throw ( CMSException )

*Gets the delivery mode for this Producer.*

- virtual void **setDisableMessageID** (bool value)=0 throw ( CMSException )

*Sets if **Message** (p. 2614) Ids are disabled for this Producer.*

- virtual bool **getDisableMessageID** () const =0 throw ( CMSException )

*Gets if **Message** (p. 2614) Ids are disabled for this Producer.*

- virtual void **setDisableMessageTimeStamp** (bool value)=0 throw ( CMSException )

*Sets if **Message** (p. 2614) Time Stamps are disabled for this Producer.*

- virtual bool **getDisableMessageTimeStamp** () const =0 throw ( CMSException )

*Gets if **Message** (p. 2614) Time Stamps are disabled for this Producer.*

- virtual void **setPriority** (int priority)=0 throw ( CMSException )

*Sets the Priority that this Producers sends messages at.*

- virtual int **getPriority** () const =0 throw ( CMSException )

*Gets the Priority level that this producer sends messages at.*

- virtual void **setTimeToLive** (long long time)=0 throw ( CMSException )

*Sets the Time to Live that this Producers sends messages with.*

- virtual long long **getTimeToLive** () const =0 throw ( CMSException )

*Gets the Time to Live that this producer sends messages with.*

### 6.559.1 Detailed Description

A client uses a **MessageProducer** (p. 2809) object to send messages to a **Destination** (p. 1776). A **MessageProducer** (p. 2809) object is created by passing a **Destination** (p. 1776) object to a message-producer creation method supplied by a session.

A client also has the option of creating a message producer without supplying a destination. In this case, a **Destination** (p. 1776) must be provided with every send operation. A typical use for this kind of message producer is to send replies to requests using the request's CMSReplyTo destination.

A client can specify a default delivery mode, priority, and time to live for messages sent by a message producer. It can also specify the delivery mode, priority, and time to live for an individual message.

A client can specify a time-to-live value in milliseconds for each message it sends. This value defines a message expiration time that is the sum of the message's time-to-live and the GMT when it is sent (for transacted sends, this is the time the client sends the message, not the time the transaction is committed).

#### Since

1.0

### 6.559.2 Constructor & Destructor Documentation

6.559.2.1 `virtual cms::MessageProducer::~MessageProducer ( ) [inline, virtual]`

### 6.559.3 Member Function Documentation

6.559.3.1 `virtual int cms::MessageProducer::getDeliveryMode ( ) const throw ( CMSEException ) [pure virtual]`

Gets the delivery mode for this Producer.

#### Returns

The **DeliveryMode** (p. 1775)

#### Exceptions

<b>CMSEException</b> (p. 1190)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedProducer** (p. 1103), and **activemq::core::ActiveMQProducer** (p. 469).

**6.559.3.2** `virtual bool cms::MessageProducer::getDisableMessageID ( ) const throw ( CMSEException ) [pure virtual]`

Gets if **Message** (p. 2614) Ids are disabled for this Producer.

#### Returns

boolean indicating enable / disable (true / false)

#### Exceptions

<b>CMSEException</b> (p. 1190)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedProducer** (p. 1103), and **activemq::core::ActiveMQProducer** (p. 469).

**6.559.3.3** `virtual bool cms::MessageProducer::getDisableMessageTimeStamp ( ) const throw ( CMSEException ) [pure virtual]`

Gets if **Message** (p. 2614) Time Stamps are disabled for this Producer.

#### Returns

boolean indicating enable / disable (true / false)

#### Exceptions

<b>CMSEException</b> (p. 1190)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedProducer** (p. 1104), and **activemq::core::ActiveMQProducer** (p. 469).

**6.559.3.4** `virtual int cms::MessageProducer::getPriority ( ) const throw ( CMSEException ) [pure virtual]`

Gets the Priority level that this producer sends messages at.

#### Returns

int based priority level

#### Exceptions

<b>CMSEException</b> (p. 1190)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in `activemq::cmsutil::CachedProducer` (p. 1104), and `activemq::core::ActiveMQProducer` (p. 469).

**6.559.3.5** `virtual long long cms::MessageProducer::getTimeToLive ( ) const throw ( CMSEException ) [pure virtual]`

Gets the Time to Live that this producer sends messages with.

#### Returns

Time to live value in milliseconds

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::cmsutil::CachedProducer` (p. 1104), and `activemq::core::ActiveMQProducer` (p. 470).

**6.559.3.6** `virtual void cms::MessageProducer::send ( Message * message, int deliveryMode, int priority, long long timeToLive ) throw ( cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException ) [pure virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

#### Parameters

<i>message</i>	The message to be sent.
<i>delivery-Mode</i>	The delivery mode to be used.
<i>priority</i>	The priority for this message.
<i>timeToLive</i>	The time to live value for this message in milliseconds.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if an internal error occurs while sending the message.
<b><i>MessageFormatException</i></b> (p. 2749)	- if an Invalid <b>Message</b> (p. 2614) is given.
<b><i>InvalidDestinationException</i></b> (p. 2200)	- if a client uses this method with a <b>MessageProducer</b> (p. 2809) with an invalid destination.
<b><i>UnsupportedOperationException</i></b> (p. 4030)	- if a client uses this method with a <b>MessageProducer</b> (p. 2809) that did not specify a destination at creation time.



Implemented in `activemq::cmsutil::CachedProducer` (p. 1105), and `activemq::core::ActiveMQProducer` (p. 472).

**6.559.3.7** `virtual void cms::MessageProducer::send ( const Destination  
* destination, Message * message, int deliveryMode, int  
priority, long long timeToLive ) throw ( cms::CMSException,  
cms::MessageFormatException, cms::InvalidDestinationException,  
cms::UnsupportedOperationException ) [pure virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

#### Parameters

<i>destination</i>	The destination on which to send the message
<i>message</i>	The message to be sent.
<i>delivery-Mode</i>	The delivery mode to be used.
<i>priority</i>	The priority for this message.
<i>timeToLive</i>	The time to live value for this message in milliseconds.

#### Exceptions

<b>CMSException</b> (p. 1190)	- if an internal error occurs while sending the message.
<b>MessageFormatException</b> (p. 2749)	- if an Invalid <b>Message</b> (p. 2614) is given.
<b>InvalidDestinationException</b> (p. 2200)	- if a client uses this method with a <b>MessageProducer</b> (p. 2809) with an invalid destination.
<b>UnsupportedOperationException</b> (p. 4030)	- if a client uses this method with a <b>MessageProducer</b> (p. 2809) that did not specify a destination at creation time.

Implemented in `activemq::cmsutil::CachedProducer` (p. 1106), and `activemq::core::ActiveMQProducer` (p. 471).

**6.559.3.8** `virtual void cms::MessageProducer::send ( Message  
* message ) throw ( cms::CMSException,  
cms::MessageFormatException, cms::InvalidDestinationException,  
cms::UnsupportedOperationException ) [pure virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

**Parameters**

<i>message</i>	The message to be sent.
----------------	-------------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if an internal error occurs while sending the message.
<b><i>MessageFormatException</i></b> (p. 2749)	- if an Invalid <b>Message</b> (p. 2614) is given.
<b><i>InvalidDestinationException</i></b> (p. 2200)	- if a client uses this method with a <b>MessageProducer</b> (p. 2809) with an invalid destination.
<b><i>UnsupportedOperationException</i></b> (p. 4030)	- if a client uses this method with a <b>MessageProducer</b> (p. 2809) that did not specify a destination at creation time.

Implemented in **activemq::cmsutil::CachedProducer** (p. 1106), and **activemq::core::ActiveMQProducer** (p. 471).

**6.559.3.9** `virtual void cms::MessageProducer::send ( const Destination *  
destination, Message * message ) throw ( cms::CMSEException,  
cms::MessageFormatException, cms::InvalidDestinationException,  
cms::UnsupportedOperationException ) [pure virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

**Parameters**

<i>destination</i>	The destination on which to send the message
<i>message</i>	the message to be sent.

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if an internal error occurs while sending the message.
<b><i>MessageFormatException</i></b> (p. 2749)	- if an Invalid <b>Message</b> (p. 2614) is given.
<b><i>InvalidDestinationException</i></b> (p. 2200)	- if a client uses this method with a <b>MessageProducer</b> (p. 2809) with an invalid destination.
<b><i>UnsupportedOperationException</i></b> (p. 4030)	- if a client uses this method with a <b>MessageProducer</b> (p. 2809) that did not specify a destination at creation time.

Implemented in **activemq::cmsutil::CachedProducer** (p. 1105), and **activemq::core::ActiveMQProducer**

(p. 472).

**6.559.3.10** `virtual void cms::MessageProducer::setDeliveryMode ( int mode ) throw ( CMSEException ) [pure virtual]`

Sets the delivery mode for this Producer.

#### Parameters

<i>mode</i>	The <b>DeliveryMode</b> (p. 1775)
-------------	-----------------------------------

#### Exceptions

<b>CMSEException</b> (p. 1190)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedProducer** (p. 1107), and **activemq::core::ActiveMQProducer** (p. 473).

**6.559.3.11** `virtual void cms::MessageProducer::setDisableMessageID ( bool value ) throw ( CMSEException ) [pure virtual]`

Sets if **Message** (p. 2614) Ids are disabled for this Producer.

#### Parameters

<i>value</i>	boolean indicating enable / disable (true / false)
--------------	--

#### Exceptions

<b>CMSEException</b> (p. 1190)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedProducer** (p. 1107), and **activemq::core::ActiveMQProducer** (p. 473).

**6.559.3.12** `virtual void cms::MessageProducer::setDisableMessageTimeStamp ( bool value ) throw ( CMSEException ) [pure virtual]`

Sets if **Message** (p. 2614) Time Stamps are disabled for this Producer.

#### Parameters

<i>value</i>	- boolean indicating enable / disable (true / false)
--------------	--

#### Exceptions

<b>CMSEException</b> (p. 1190)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedProducer** (p. 1107), and **activemq::core::ActiveMQProducer** (p. 473).

**6.559.3.13** `virtual void cms::MessageProducer::setPriority ( int priority ) throw ( CMSEException )` [pure virtual]

Sets the Priority that this Producers sends messages at.

#### Parameters

<i>priority</i>	int value for Priority level
-----------------	------------------------------

#### Exceptions

<b>CMSEException</b> (p. 1190)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedProducer** (p. 1108), and **activemq::core::ActiveMQProducer** (p. 474).

**6.559.3.14** `virtual void cms::MessageProducer::setTimeToLive ( long long time ) throw ( CMSEException )` [pure virtual]

Sets the Time to Live that this Producers sends messages with.

This value will be used if the time to live is not specified via the send method.

#### Parameters

<i>time</i>	default time to live value in milliseconds
-------------	--

#### Exceptions

<b>CMSEException</b> (p. 1190)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedProducer** (p. 1108), and **activemq::core::ActiveMQProducer** (p. 474).

The documentation for this class was generated from the following file:

- `src/main/cms/MessageProducer.h`

## 6.560 **activemq::wireformat::openwire::utils::MessagePropertyInterceptor** Class Reference

Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties.

```
#include <src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h>
```

## Public Member Functions

- **MessagePropertyInterceptor** (**commands::Message** \*message, **util::PrimitiveMap** \*properties) throw ( **decaf::lang::exceptions::NullPointerException** )

*Constructor, accepts the Message that will be used to store JMS reserved property values, and the PrimitiveMap to get and set the rest to.*

- virtual **~MessagePropertyInterceptor** ()
- virtual bool **getBooleanProperty** (const std::string &name) const  
*Gets a boolean property.*
- virtual unsigned char **getByteProperty** (const std::string &name) const  
*Gets a byte property.*
- virtual double **getDoubleProperty** (const std::string &name) const  
*Gets a double property.*
- virtual float **getFloatProperty** (const std::string &name) const  
*Gets a float property.*
- virtual int **getIntProperty** (const std::string &name) const  
*Gets a int property.*
- virtual long long **getLongProperty** (const std::string &name) const  
*Gets a long property.*
- virtual short **getShortProperty** (const std::string &name) const  
*Gets a short property.*
- virtual std::string **getStringProperty** (const std::string &name) const  
*Gets a string property.*
- virtual void **setBooleanProperty** (const std::string &name, bool value)  
*Sets a boolean property.*
- virtual void **setByteProperty** (const std::string &name, unsigned char value)  
*Sets a byte property.*
- virtual void **setDoubleProperty** (const std::string &name, double value)  
*Sets a double property.*
- virtual void **setFloatProperty** (const std::string &name, float value)  
*Sets a float property.*

- virtual void **setIntProperty** (const std::string &name, int value)  
*Sets a int property.*
- virtual void **setLongProperty** (const std::string &name, long long value)  
*Sets a long property.*
- virtual void **setShortProperty** (const std::string &name, short value)  
*Sets a short property.*
- virtual void **setStringProperty** (const std::string &name, const std::string &value)  
*Sets a string property.*

### 6.560.1 Detailed Description

Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties. Currently the only properties that are intercepted and handled are:

Name | Conversion Supported ----- JMSXDe-  
liveryCount | Int, Long, String JMSXGroupID | String JMSXGroupSeq | Int, Long,  
String

### 6.560.2 Constructor & Destructor Documentation

**6.560.2.1** `activemq::wireformat::openwire::utils::MessagePropertyInterceptor::MessagePropertyInterceptor ( commands::Message * message, util::PrimitiveMap * properties ) throw ( decaf::lang::exceptions::NullPointerException )`

Constructor, accepts the Message that will be used to store JMS reserved property values, and the PrimitiveMap to get and set the rest to.

#### Parameters

<i>message</i>	- The Message to store reserved property data in
<i>properties</i>	- The PrimitiveMap to store the rest of the properties in.

#### Exceptions

<i>NullPointerException</i>	if either param is NULL
-----------------------------	-------------------------

## 6.560 activemq::wireformat::openwire::utils::MessagePropertyInterceptor Class Reference 2823

6.560.2.2 virtual activemq::wireformat::openwire::utils::MessagePropertyInterceptor::~MessagePropertyInterceptor ( ) [virtual]

### 6.560.3 Member Function Documentation

6.560.3.1 virtual bool activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getBooleanProperty ( const std::string & *name* ) const [virtual]

Gets a boolean property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

6.560.3.2 virtual unsigned char activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getBytesProperty ( const std::string & *name* ) const [virtual]

Gets a byte property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

6.560.3.3 virtual double activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getDoubleProperty ( const std::string & *name* ) const [virtual]

Gets a double property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

6.560.3.4 virtual float activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getFloatProperty ( const std::string & *name* ) const [virtual]

Gets a float property.

**Parameters**

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

**Returns**

The value for the named property.

**6.560.3.5** `virtual int activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getIntProperty  
( const std::string & name ) const` [virtual]

Gets a int property.

**Parameters**

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

**Returns**

The value for the named property.

**6.560.3.6** `virtual long long activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getLongProperty  
( const std::string & name ) const` [virtual]

Gets a long property.

**Parameters**

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

**Returns**

The value for the named property.

**6.560.3.7** `virtual short activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getShortProperty  
( const std::string & name ) const` [virtual]

Gets a short property.

**Parameters**

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

**Returns**

The value for the named property.



## 6.560 activemq::wireformat::openwire::utils::MessagePropertyInterceptor Class Reference 2825

**6.560.3.8** virtual std::string activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getStringProperty ( const std::string & *name* ) const [virtual]

Gets a string property.

### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

### Returns

The value for the named property.

**6.560.3.9** virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setBooleanProperty ( const std::string & *name*, bool *value* ) [virtual]

Sets a boolean property.

### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

**6.560.3.10** virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setByteProperty ( const std::string & *name*, unsigned char *value* ) [virtual]

Sets a byte property.

### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

**6.560.3.11** virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setDoubleProperty ( const std::string & *name*, double *value* ) [virtual]

Sets a double property.

### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

**6.560.3.12** `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setFloatProperty  
( const std::string & name, float value ) [virtual]`

Sets a float property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

**6.560.3.13** `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setIntProperty  
( const std::string & name, int value ) [virtual]`

Sets a int property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

**6.560.3.14** `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setLongProperty  
( const std::string & name, long long value ) [virtual]`

Sets a long property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

**6.560.3.15** `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setShortProperty  
( const std::string & name, short value ) [virtual]`

Sets a short property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

**6.560.3.16** `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setStringProperty  
( const std::string & name, const std::string & value ) [virtual]`

Sets a string property.

**Parameters**

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/utis/**MessagePropertyInterceptor.h**

**6.561 activemq::commands::MessagePull Class Reference**

```
#include <src/main/activemq/commands/MessagePull.h>
```

Inheritance diagram for activemq::commands::MessagePull:

**Public Member Functions**

- **MessagePull ()**
- virtual **~MessagePull ()**
- virtual unsigned char **getDataStructureType ()** const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **MessagePull \* cloneDataStructure ()** const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure (const DataStructure \*src)**  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString ()** const  
*Returns a string containing the information for this DataStructure (p. 1713) such as its type and value of its elements.*
- virtual bool **equals (const DataStructure \*value)** const  
*Compares the DataStructure (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer< ConsumerId > &getConsumerId ()** const
- virtual **Pointer< ConsumerId > &getConsumerId ()**
- virtual void **setConsumerId (const Pointer< ConsumerId > &consumerId)**
- virtual const **Pointer< ActiveMQDestination > &getDestination ()** const
- virtual **Pointer< ActiveMQDestination > &getDestination ()**
- virtual void **setDestination (const Pointer< ActiveMQDestination > &destination)**

- virtual long long **getTimeout** () const
- virtual void **setTimeout** (long long **timeout**)
- virtual const std::string & **getCorrelationId** () const
- virtual std::string & **getCorrelationId** ()
- virtual void **setCorrelationId** (const std::string &**correlationId**)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &**messageId**)
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** \***visitor**)  
throw ( **exceptions::ActiveMQException** )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_MESSAGEPULL** = 20

### Protected Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- long long **timeout**
- std::string **correlationId**
- **Pointer**< **MessageId** > **messageId**

## 6.561.1 Constructor & Destructor Documentation

6.561.1.1 **activemq::commands::MessagePull::MessagePull** ( )

6.561.1.2 **virtual activemq::commands::MessagePull::~~MessagePull** ( ) [virtual]

## 6.561.2 Member Function Documentation

6.561.2.1 **virtual MessagePull\*** **activemq::commands::MessagePull::cloneDataStructure** ( )  
**const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

**6.561.2.2** `virtual void activemq::commands::MessagePull::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

**6.561.2.3** `virtual bool activemq::commands::MessagePull::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

**6.561.2.4** `virtual const Pointer<ConsumerId>& activemq::commands::MessagePull::getConsumerId ( ) const [virtual]`

**6.561.2.5** `virtual Pointer<ConsumerId>& activemq::commands::MessagePull::getConsumerId ( ) [virtual]`

**6.561.2.6** `virtual const std::string& activemq::commands::MessagePull::getCorrelationId ( ) const [virtual]`

**6.561.2.7** `virtual std::string& activemq::commands::MessagePull::getCorrelationId ( ) [virtual]`

**6.561.2.8** `virtual unsigned char activemq::commands::MessagePull::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

- 6.561.2.9 **virtual const Pointer<ActiveMQDestination>&**  
**activemq::commands::MessagePull::getDestination ( ) const** [virtual]
  
- 6.561.2.10 **virtual Pointer<ActiveMQDestination>&**  
**activemq::commands::MessagePull::getDestination ( )** [virtual]
  
- 6.561.2.11 **virtual const Pointer<MessageId>& ac-**  
**tivemq::commands::MessagePull::getMessageId ( ) const**  
 [virtual]
  
- 6.561.2.12 **virtual Pointer<MessageId>& ac-**  
**tivemq::commands::MessagePull::getMessageId ( )**  
 [virtual]
  
- 6.561.2.13 **virtual long long activemq::commands::MessagePull::getTimeout ( ) const**  
 [virtual]
  
- 6.561.2.14 **virtual void activemq::commands::MessagePull::setConsumerId ( const Pointer<**  
**ConsumerId > & consumerId )** [virtual]
  
- 6.561.2.15 **virtual void activemq::commands::MessagePull::setCorrelationId ( const std::string**  
**& correlationId )** [virtual]
  
- 6.561.2.16 **virtual void activemq::commands::MessagePull::setDestination ( const Pointer<**  
**ActiveMQDestination > & destination )** [virtual]
  
- 6.561.2.17 **virtual void activemq::commands::MessagePull::setMessageId ( const Pointer<**  
**MessageId > & messageId )** [virtual]
  
- 6.561.2.18 **virtual void activemq::commands::MessagePull::setTimeout ( long long timeout )**  
 [virtual]
  
- 6.561.2.19 **virtual std::string activemq::commands::MessagePull::toString ( ) const**  
 [virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

6.561.2.20 `virtual Pointer<Command> activemq::commands::MessagePull::visit  
( activemq::state::CommandVisitor * visitor ) throw (  
exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

#### 6.561.3 Field Documentation

6.561.3.1 `Pointer<ConsumerId> activemq::commands::MessagePull::consumerId  
[protected]`

6.561.3.2 `std::string activemq::commands::MessagePull::correlationId  
[protected]`

6.561.3.3 `Pointer<ActiveMQDestination> ac-  
tivemq::commands::MessagePull::destination  
[protected]`

6.561.3.4 `const unsigned char activemq::commands::MessagePull::ID_-  
MESSAGEPULL = 20 [static]`

6.561.3.5 `Pointer<MessageId> activemq::commands::MessagePull::messageId  
[protected]`

6.561.3.6 `long long activemq::commands::MessagePull::timeout [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessagePull.h`

## 6.562 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2828).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller:

## Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.562.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2828). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.562.2 Constructor & Destructor Documentation

6.562.2.1 `activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::MessagePullMarshaller ( ) [inline]`

6.562.2.2 `virtual activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::~~MessagePullMarshaller ( ) [inline, virtual]`

## 6.562.3 Member Function Documentation

6.562.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.562.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.562.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 808).

**6.562.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 809).

**6.562.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 810).

**6.562.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 811).

**6.562.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 813).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h`

## 6.563 `activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2833).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller`:

### Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.563.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2833). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.563.2 Constructor & Destructor Documentation

- 6.563.2.1 **activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::MessagePullMarshaller**  
( ) [inline]
- 6.563.2.2 **virtual activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::~~MessagePullMarshaller**  
( ) [inline, virtual]

### 6.563.3 Member Function Documentation

- 6.563.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

- 6.563.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

- 6.563.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 794).

```
6.563.3.4  virtual void activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 795).

```
6.563.3.5  virtual int activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 796).

**6.563.3.6** virtual void activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::tightMarshal2  
( OpenWireFormat \* *wireFormat*, commands::DataStructure  
\* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 797).

**6.563.3.7** virtual void activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::tightUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream  
\* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h`

## 6.564 **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller** Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2837).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller**:

**Public Member Functions**

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*



- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.564.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2837). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.564.2 Constructor & Destructor Documentation

6.564.2.1 **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::MessagePullMarshaller**  
( ) [inline]

6.564.2.2 **virtual activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::~~MessagePullMarshaller**  
( ) [inline, virtual]

### 6.564.3 Member Function Documentation

6.564.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.564.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

```
6.564.3.3 virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 772).

```
6.564.3.4 virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 774).

6.564.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 775).

6.564.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 776).

```
6.564.3.7 virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h

## 6.565 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2841).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller**:

#### Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.565.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2841). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.565.2 Constructor & Destructor Documentation

**6.565.2.1** **activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::MessagePullMarshaller**  
( ) [inline]

**6.565.2.2** **virtual activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::~~MessagePullMarshaller**  
( ) [inline, virtual]

### 6.565.3 Member Function Documentation

**6.565.3.1** **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.565.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.565.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 780).

**6.565.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 781).

**6.565.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 782).

**6.565.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 783).

**6.565.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h`

## 6.566 **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller** Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2845).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller**:



## Public Member Functions

- **MessagePullMarshaller ()**
- virtual **~MessagePullMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.566.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2845). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.566.2 Constructor & Destructor Documentation

6.566.2.1 `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::MessagePullMarshaller ( )` `[inline]`

6.566.2.2 `virtual activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::~~MessagePullMarshaller ( )` `[inline, virtual]`

## 6.566.3 Member Function Documentation

6.566.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::createObject ( )`  
`const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.566.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::getDataStructureType ( )` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.566.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut )` `throw ( decaf::io::IOException )` `[virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 787).

**6.566.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 788).

**6.566.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 789).

```
6.566.3.6  virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 790).

```
6.566.3.7  virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 792).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h

## 6.567 activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2850).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/MessagePullMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller:

### Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.567.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2850). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.567.2 Constructor & Destructor Documentation

6.567.2.1 **activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::MessagePullMarshaller**  
( ) [inline]

6.567.2.2 **virtual activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::~~MessagePullMarshaller**  
( ) [inline, virtual]

### 6.567.3 Member Function Documentation

6.567.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.567.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.567.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 801).

**6.567.3.4** virtual void activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::looseUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 802).

**6.567.3.5** virtual int activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::tightMarshal1 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 803).

```
6.567.3.6 virtual void activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 804).

```
6.567.3.7 virtual void activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.



## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 806).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/MessagePullMarshaller.h

## 6.568 activemq::transport::mock::MockTransport Class Reference

The **MockTransport** (p. 2854) defines a base level **Transport** (p. 3996) class that is intended to be used in place of an a regular protocol **Transport** (p. 3996) such as TCP.

```
#include <src/main/activemq/transport/mock/MockTransport.h>
```

Inheritance diagram for **activemq::transport::mock::MockTransport**:

## Public Member Functions

- **MockTransport** (const **Pointer**< **wireformat::WireFormat** > &wireFormat, const **Pointer**< **ResponseBuilder** > &responseBuilder)
- virtual ~**MockTransport** ()
- void **setResponseBuilder** (const **Pointer**< **ResponseBuilder** > &responseBuilder)

*Sets the **ResponseBuilder** (p. 3382) that this class uses to create Responses to Commands sent.*

- virtual void **oneway** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

*Sends a one-way command.*

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

*Sends the given command to the broker and then waits for the response.*

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

*Sends the given command to the broker and then waits for the response.*

- virtual void **setOutgoingListener** (**TransportListener** \*listener)

*Sets a Listener that gets notified for every command that would have been sent by this transport to the Broker, this allows a client to verify that its messages are making it to the wire.*

- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wire-Format AMQCPP\_UNUSED)

*Sets the WireFormat instance to use.*

- **Pointer**< **wireformat::WireFormat** > **getWireFormat** () const

*Gets the currently set WireFormat.*

- virtual void **setTransportListener** (**TransportListener** \*listener)

*Sets the observer of asynchronous exceptions from this transport.*

- virtual **TransportListener** \* **getTransportListener** () const

*Gets the observer of asynchronous exceptions from this transport.*

- virtual void **fireCommand** (const **Pointer**< **Command** > &command)

*Fires a Command back through this transport to its registered CommandListener if there is one.*

- virtual void **fireException** (const **exceptions::ActiveMQException** &ex)

*Fires a Exception back through this transport to its registered ExceptionListener if there is one.*

- virtual void **start** () throw ( decaf::io::IOException )

*Starts the **Transport** (p. 3996), the send methods of a **Transport** (p. 3996) will throw an exception if used before the **Transport** (p. 3996) is started.*

- virtual void **stop** () throw ( decaf::io::IOException )

*Stops the **Transport** (p. 3996).*

- virtual void **close** () throw ( decaf::io::IOException )

*Closes this object and deallocates the appropriate resources.*

- virtual **Transport** \* **narrow** (const std::type\_info &typeId)

*Narrows down a Chain of Transports to a specific **Transport** (p. 3996) to allow a higher level transport to skip intermediate Transports in certain circumstances.*

- virtual bool **isFaultTolerant** () const

*Is this **Transport** (p. 3996) fault tolerant, meaning that it will reconnect to a broker on disconnect.*

- virtual bool **isConnected** () const

*Is the **Transport** (p. 3996) Connected to its Broker.*

- virtual bool **isClosed** () const

*Has the **Transport** (p. 3996) been shutdown and no longer usable.*

- virtual std::string **getRemoteAddress** () const
- virtual void **reconnect** (const **decaf::net::URI** &uri AMQCPP\_UNUSED) throw ( decaf::io::IOException )  
*reconnect to another location*
- bool **isFailOnSendMessage** () const
- void **setFailOnSendMessage** (bool value)
- int **getNumSendMessageBeforeFail** () const
- void **setNumSendMessageBeforeFail** (int value)
- int **getNumSentMessages** () const
- void **setNumSentMessages** (int value)
- bool **isFailOnReceiveMessage** () const
- void **setFailOnReceiveMessage** (bool value)
- int **getNumReceivedMessageBeforeFail** () const
- void **setNumReceivedMessageBeforeFail** (int value)
- int **getNumReceivedMessages** () const
- void **setNumReceivedMessages** (int value)
- bool **isFailOnKeepAliveSends** () const
- void **setFailOnKeepAliveSends** (bool value)
- int **getNumSentKeepAlivesBeforeFail** () const
- void **setNumSentKeepAlivesBeforeFail** (int value)
- int **getNumSentKeepAlives** () const
- void **setNumSentKeepAlives** (int value)
- bool **isFailOnStart** () const
- void **setFailOnStart** (bool value)
- bool **isFailOnStop** () const
- void **setFailOnStop** (bool value)
- bool **isFailOnClose** () const
- void **setFailOnClose** (bool value)

### Static Public Member Functions

- static **MockTransport** \* **getInstance** ()

#### 6.568.1 Detailed Description

The **MockTransport** (p. 2854) defines a base level **Transport** (p. 3996) class that is intended to be used in place of an regular protocol **Transport** (p. 3996) such as TCP. This **Transport** (p. 3996) assumes that it is the base **Transport** (p. 3996) in the Transports stack, and destroys any Transports that are passed to it in its constructor.

This **Transport** (p. 3996) defines an Interface **ResponseBuilder** (p. 3382) which must be implemented by any protocol for which the **Transport** (p. 3996) is used to Emulate. The **Transport** (p. 3996) hands off all outbound commands to the **ResponseBuilder** (p. 3382) for processing, it is up to the builder to create appropriate responses and schedule any asynchronous messages that might result from a message sent to the Broker.

## 6.568.2 Constructor & Destructor Documentation

6.568.2.1 `activemq::transport::mock::MockTransport::MockTransport ( const Pointer< wireformat::WireFormat > & wireFormat, const Pointer< ResponseBuilder > & responseBuilder )`

6.568.2.2 `virtual activemq::transport::mock::MockTransport::~~MockTransport ( )`  
[inline, virtual]

## 6.568.3 Member Function Documentation

6.568.3.1 `virtual void activemq::transport::mock::MockTransport::close ( ) throw ( decaf::io::IOException )` [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

### Exceptions

<i>IOException</i>	if an error occurs while closing.
--------------------	-----------------------------------

Implements `decaf::io::Closeable` (p. 1181).

6.568.3.2 `virtual void activemq::transport::mock::MockTransport::fireCommand ( const Pointer< Command > & command )` [inline, virtual]

Fires a Command back through this transport to its registered CommandListener if there is one.

### Parameters

<i>command</i>	- Command to send to the Listener.
----------------	------------------------------------

6.568.3.3 `virtual void activemq::transport::mock::MockTransport::fireException ( const exceptions::ActiveMQException & ex )` [inline, virtual]

Fires a Exception back through this transport to its registered ExceptionListener if there is one.

### Parameters

<i>ex</i>	The Exception that will be passed on the the <b>Transport</b> (p. 3996) listener.
-----------	---

6.568.3.4 **static MockTransport\*** **activemq::transport::mock::MockTransport::getInstance (**  
    **)** **[inline, static]**

6.568.3.5 **int** **activemq::transport::mock::MockTransport::getNumReceivedMessageBeforeFail (**  
    **)** **const** **[inline]**

6.568.3.6 **int** **activemq::transport::mock::MockTransport::getNumReceivedMessages ( )** **const**  
    **[inline]**

6.568.3.7 **int** **activemq::transport::mock::MockTransport::getNumSentKeepAlives ( )** **const**  
    **[inline]**

6.568.3.8 **int** **activemq::transport::mock::MockTransport::getNumSentKeepAlivesBeforeFail ( )**  
    **const** **[inline]**

6.568.3.9 **int** **activemq::transport::mock::MockTransport::getNumSentMessageBeforeFail ( )**  
    **const** **[inline]**

6.568.3.10 **int** **activemq::transport::mock::MockTransport::getNumSentMessages ( )** **const**  
    **[inline]**

6.568.3.11 **virtual std::string** **activemq::transport::mock::MockTransport::getRemoteAddress (**  
    **)** **const** **[inline, virtual]**

#### Returns

the remote address for this connection

Implements **activemq::transport::Transport** (p. 3998).

6.568.3.12 **virtual TransportListener\*** **ac-**  
    **tivemq::transport::mock::MockTransport::getTransportListener (**  
        **)** **const** **[inline, virtual]**

Gets the observer of asynchronous exceptions from this transport.

#### Returns

The listener of transport events.

Implements **activemq::transport::Transport** (p. 3998).

6.568.3.13 **Pointer<wireformat::WireFormat>** **ac-**  
    **tivemq::transport::mock::MockTransport::getWireFormat ( )** **const**  
        **[inline]**

Gets the currently set WireFormat.

**Returns**

the current WireFormat object.

**6.568.3.14** `virtual bool activemq::transport::mock::MockTransport::isClosed ( ) const`  
[inline, virtual]

Has the **Transport** (p. 3996) been shutdown and no longer usable.

**Returns**

true if the **Transport** (p. 3996)

Implements **activemq::transport::Transport** (p. 3998).

**6.568.3.15** `virtual bool activemq::transport::mock::MockTransport::isConnected ( ) const`  
[inline, virtual]

Is the **Transport** (p. 3996) Connected to its Broker.

**Returns**

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 3998).

**6.568.3.16** `bool activemq::transport::mock::MockTransport::isFailOnClose ( ) const`  
[inline]

**6.568.3.17** `bool activemq::transport::mock::MockTransport::isFailOnKeepAliveSends ( ) const`  
[inline]

**6.568.3.18** `bool activemq::transport::mock::MockTransport::isFailOnReceiveMessage ( ) const`  
[inline]

**6.568.3.19** `bool activemq::transport::mock::MockTransport::isFailOnSendMessage ( ) const`  
[inline]

**6.568.3.20** `bool activemq::transport::mock::MockTransport::isFailOnStart ( ) const`  
[inline]

**6.568.3.21** `bool activemq::transport::mock::MockTransport::isFailOnStop ( ) const`  
[inline]

**6.568.3.22** `virtual bool activemq::transport::mock::MockTransport::isFaultTolerant ( ) const`  
[inline, virtual]

Is this **Transport** (p. 3996) fault tolerant, meaning that it will reconnect to a broker on disconnect.

**Returns**

true if the **Transport** (p. 3996) is fault tolerant.

Implements **activemq::transport::Transport** (p. 3999).

**6.568.3.23** `virtual Transport* activemq::transport::mock::MockTransport::narrow ( const std::type_info & typeId ) [inline, virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 3996) to allow a higher level transport to skip intermediate Transports in certain circumstances.

**Parameters**

<i>typeId</i>	- The type_info of the Object we are searching for.
---------------	---

**Returns**

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 3999).

**6.568.3.24** `virtual void activemq::transport::mock::MockTransport::oneway ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

**Parameters**

<i>command</i>	the command to be sent.
----------------	-------------------------

**Exceptions**

<i>IOException</i>	if an exception occurs during writing of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 3999).

**6.568.3.25** `virtual void activemq::transport::mock::MockTransport::reconnect ( const decaf::net::URI &uri AMQCPP_UNUSED ) throw ( decaf::io::IOException) [inline, virtual]`

reconnect to another location

**Parameters**

<i>uri</i>	
------------	--

**Exceptions**

<i>IOException</i>	on failure of if not supported
--------------------	--------------------------------

**6.568.3.26** `virtual Pointer<Response> activemq::transport::mock::MockTransport::request ( const Pointer< Command > & command, unsigned int timeout ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)`  
[virtual]

Sends the given command to the broker and then waits for the response.

**Parameters**

<i>command</i>	- The command to be sent.
<i>timeout</i>	- The time to wait for this response.

**Returns**

the response from the broker.

**Exceptions**

<i>IOException</i>	if an exception occurs during the read of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implements `activemq::transport::Transport` (p. 4001).

**6.568.3.27** `virtual Pointer<Response> activemq::transport::mock::MockTransport::request ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)`  
[virtual]

Sends the given command to the broker and then waits for the response.

**Parameters**

<i>command</i>	the command to be sent.
----------------	-------------------------

**Returns**

the response from the broker.

**Exceptions**

<i>IOException</i>	if an exception occurs during the read of the command.
--------------------	--



<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.
--------------------------------------	--

Implements **activemq::transport::Transport** (p. 4000).

- 6.568.3.28 **void** activemq::transport::mock::MockTransport::setFailOnClose ( *bool value* )  
[inline]
- 6.568.3.29 **void** activemq::transport::mock::MockTransport::setFailOnKeepAliveSends ( *bool value* ) [inline]
- 6.568.3.30 **void** activemq::transport::mock::MockTransport::setFailOnReceiveMessage ( *bool value* ) [inline]
- 6.568.3.31 **void** activemq::transport::mock::MockTransport::setFailOnSendMessage ( *bool value* ) [inline]
- 6.568.3.32 **void** activemq::transport::mock::MockTransport::setFailOnStart ( *bool value* )  
[inline]
- 6.568.3.33 **void** activemq::transport::mock::MockTransport::setFailOnStop ( *bool value* )  
[inline]
- 6.568.3.34 **void** activemq::transport::mock::MockTransport::setNumReceivedMessageBeforeFail  
( *int value* ) [inline]
- 6.568.3.35 **void** activemq::transport::mock::MockTransport::setNumReceivedMessages ( *int value* ) [inline]
- 6.568.3.36 **void** activemq::transport::mock::MockTransport::setNumSentKeepAlives ( *int value* )  
[inline]
- 6.568.3.37 **void** activemq::transport::mock::MockTransport::setNumSentKeepAlivesBeforeFail ( *int value* ) [inline]
- 6.568.3.38 **void** activemq::transport::mock::MockTransport::setNumSentMessageBeforeFail ( *int value* ) [inline]
- 6.568.3.39 **void** activemq::transport::mock::MockTransport::setNumSentMessages ( *int value* )  
[inline]
- 6.568.3.40 **virtual void** activemq::transport::mock::MockTransport::setOutgoingListener ( *TransportListener \* listener* ) [inline, virtual]

Sets a Listener that gets notified for every command that would have been sent by this transport to the Broker, this allows a client to verify that its messages are making it to the wire.

**Parameters**

<i>listener</i>	- The <b>CommandListener</b> to notify for each message
-----------------	---

**6.568.3.41** `void activemq::transport::mock::MockTransport::setResponseBuilder ( const Pointer< ResponseBuilder > & responseBuilder ) [inline]`

Sets the **ResponseBuilder** (p. 3382) that this class uses to create Responses to Commands sent.

These are either real Response Objects, or Commands that would have been sent Asynchronously by the Broker.

**Parameters**

<i>response-Builder</i>	- The <b>ResponseBuilder</b> (p. 3382) to use from now on.
-------------------------	--

**6.568.3.42** `virtual void activemq::transport::mock::MockTransport::setTransportListener ( TransportListener * listener ) [inline, virtual]`

Sets the observer of asynchronous exceptions from this transport.

**Parameters**

<i>listener</i>	the listener of transport events.
-----------------	-----------------------------------

Implements **activemq::transport::Transport** (p. 4001).

**6.568.3.43** `virtual void activemq::transport::mock::MockTransport::setWireFormat ( const Pointer< wireformat::WireFormat > &wireFormat AMQCPP_UNUSED ) [inline, virtual]`

Sets the WireFormat instance to use.

**Parameters**

<i>wireFormat</i>	WireFormat the object used to encode / decode commands.
-------------------	---

**6.568.3.44** `virtual void activemq::transport::mock::MockTransport::start ( ) throw ( decaf::io::IOException ) [virtual]`

Starts the **Transport** (p. 3996), the send methods of a **Transport** (p. 3996) will throw an exception if used before the **Transport** (p. 3996) is started.

**Exceptions**

<i>IOException</i>	if and error occurs while starting the <b>Transport</b> (p. 3996).
--------------------	--

## 6.569 activemq::transport::mock::MockTransportFactory Class Reference 2867

Implements **activemq::transport::Transport** (p. 4002).

6.568.3.45 `virtual void activemq::transport::mock::MockTransport::stop ( ) throw ( decaf::io::IOException ) [virtual]`

Stops the **Transport** (p. 3996).

### Exceptions

<i>IOException</i>	if an error occurs while stopping the transport.
--------------------	--

Implements **activemq::transport::Transport** (p. 4002).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/mock/MockTransport.h`

## 6.569 activemq::transport::mock::MockTransportFactory Class Reference

Manufactures MockTransports, which are objects that read from input streams and write to output streams.

```
#include <src/main/activemq/transport/mock/MockTransportFactory.h>
```

Inheritance diagram for **activemq::transport::mock::MockTransportFactory**:

### Public Member Functions

- `virtual ~MockTransportFactory ()`
- `virtual Pointer< Transport > create (const decaf::net::URI &location) throw ( exceptions::ActiveMQException )`  
*Creates a fully configured **Transport** (p. 3996) instance which could be a chain of filters and transports.*
- `virtual Pointer< Transport > createComposite (const decaf::net::URI &location) throw ( exceptions::ActiveMQException )`  
*Creates a slimmed down **Transport** (p. 3996) instance which can be used in composite transport instances.*

### Protected Member Functions

- `virtual Pointer< Transport > doCreateComposite (const decaf::net::URI &location, const Pointer< wireformat::WireFormat > &wireFormat, const decaf::util::Properties &properties) throw ( exceptions::ActiveMQException )`

*Creates a slimed down **Transport** (p. 3996) instance which can be used in composite transport instances.*

### 6.569.1 Detailed Description

Manufactures MockTransports, which are objects that read from input streams and write to output streams.

### 6.569.2 Constructor & Destructor Documentation

6.569.2.1 `virtual activemq::transport::mock::MockTransportFactory::~~MockTransportFactory ( ) [inline, virtual]`

### 6.569.3 Member Function Documentation

6.569.3.1 `virtual Pointer<Transport> activemq::transport::mock::MockTransportFactory::create ( const decaf::net::URI & location ) throw ( exceptions::ActiveMQException ) [virtual]`

Creates a fully configured **Transport** (p. 3996) instance which could be a chain of filters and transports.

#### Parameters

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

#### Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implements `activemq::transport::TransportFactory` (p. 4003).

6.569.3.2 `virtual Pointer<Transport> activemq::transport::mock::MockTransportFactory::createComposite ( const decaf::net::URI & location ) throw ( exceptions::ActiveMQException ) [virtual]`

Creates a slimed down **Transport** (p. 3996) instance which can be used in composite transport instances.

#### Parameters

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

**Exceptions**

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implements **activemq::transport::TransportFactory** (p. 4004).

**6.569.3.3** virtual **Pointer<Transport>** **activemq::transport::mock::MockTransportFactory::doCreateComposite** ( const **decaf::net::URI** & *location*, const **Pointer<WireFormat>** & *wireFormat*, const **decaf::util::Properties** & *properties* ) throw ( **exceptions::ActiveMQException** ) [protected, virtual]

Creates a slimmed down **Transport** (p. 3996) instance which can be used in composite transport instances.

**Parameters**

<i>location</i>	- URI location to connect to.
<i>wireFormat</i>	- the assigned WireFormat for the new <b>Transport</b> (p. 3996).
<i>properties</i>	- Properties to apply to the transport.

**Returns**

Pointer to a new **Transport** (p. 3996) instance.

**Exceptions**

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

The documentation for this class was generated from the following file:

- src/main/activemq/transport/mock/**MockTransportFactory.h**

**6.570 decaf::util::concurrent::Mutex Class Reference**

**Mutex** (p. 2866) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.

```
#include <src/main/decaf/util/concurrent/Mutex.h>
```

Inheritance diagram for decaf::util::concurrent::Mutex:

**Public Member Functions**

- **Mutex** ()
- virtual **~Mutex** ()
- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )

*Locks the object.*

- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )

*Attempts to **Lock** (p. 2450) the object, if the lock is already held by another thread than this method returns false.*

- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )

*Unlocks the object.*

- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals a waiter on this object that it can now wake up and continue.*

- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals the waiters on this object that it can now wake up and continue.*

### 6.570.1 Detailed Description

**Mutex** (p. 2866) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.

**Since**

1.0

## 6.570.2 Constructor & Destructor Documentation

6.570.2.1 `decaf::util::concurrent::Mutex::Mutex ( )`

6.570.2.2 `virtual decaf::util::concurrent::Mutex::~~Mutex ( )` [virtual]

## 6.570.3 Member Function Documentation

6.570.3.1 `virtual void decaf::util::concurrent::Mutex::lock ( ) throw ( decaf::lang::exceptions::RuntimeException )` [virtual]

Locks the object.

### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3812).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::lock()`, `decaf::util::StlMap< std::string, cms::Topic * >::lock()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::lock()`, and `decaf::util::AbstractCollection< cms::Connection * >::lock()`.

6.570.3.2 `virtual void decaf::util::concurrent::Mutex::notify ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )` [virtual]

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3811) Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3813).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::notify()`, `decaf::util::StlMap< std::string, cms::Topic * >::notify()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::notify()`, and `decaf::util::AbstractCollection< cms::Connection * >::notify()`.

6.570.3.3 `virtual void decaf::util::concurrent::Mutex::notifyAll ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )` [virtual]

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3811) Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3814).

Referenced by decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::notifyAll(), decaf::util::StlMap< std::string, cms::Topic \* >::notifyAll(), decaf::util::concurrent::ConcurrentStlMap< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::notifyAll(), and decaf::util::AbstractCollection< cms::Connection \* >::notifyAll().

**6.570.3.4** `virtual bool decaf::util::concurrent::Mutex::tryLock ( ) throw ( decaf::lang::exceptions::RuntimeException )` [virtual]

Attempts to **Lock** (p. 2450) the object, if the lock is already held by another thread than this method returns false.

### Returns

true if the lock was acquired, false if it is already held by another thread.

### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3815).

Referenced by decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::tryLock(), decaf::util::StlMap< std::string, cms::Topic \* >::tryLock(), decaf::util::concurrent::ConcurrentStlMap< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::tryLock(), and decaf::util::AbstractCollection< cms::Connection \* >::tryLock().

**6.570.3.5** `virtual void decaf::util::concurrent::Mutex::unlock ( ) throw ( decaf::lang::exceptions::RuntimeException )` [virtual]

Unlocks the object.

### Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3816).

Referenced by decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::unlock(), decaf::util::StlMap< std::string, cms::Topic \* >::unlock(), decaf::util::concurrent::ConcurrentStlMap< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::unlock(),



and decaf::util::AbstractCollection< cms::Connection \* >::unlock().

**6.570.3.6** virtual void decaf::util::concurrent::Mutex::wait ( ) throw  
( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalMonitorStateException,  
decaf::lang::exceptions::InterruptedException ) [virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3811) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3818).

Referenced by decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::wait(), decaf::util::StlMap< std::string, cms::Topic \* >::wait(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::wait(), and decaf::util::AbstractCollection< cms::Connection \* >::wait().

**6.570.3.7** virtual void decaf::util::concurrent::Mutex::wait ( long long *millisecs* ) throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalMonitorStateException,  
decaf::lang::exceptions::InterruptedException ) [virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3811) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3819).

6.570.3.8 `virtual void decaf::util::concurrent::Mutex::wait ( long long millisecs, int nanos ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

#### Exceptions

<i>IllegalArgumentException</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3811) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3820).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Mutex.h`

## 6.571 decaf::util::concurrent::MutexHandle Class Reference

```
#include <src/main/decaf/internal/util/concurrent/unix/MutexHandle.h>
```

#### Public Member Functions

- **MutexHandle ()**
- **~MutexHandle ()**
- **MutexHandle ()**
- **~MutexHandle ()**

## Data Fields

- pthread\_mutex\_t **mutex**
- volatile long long **lock\_owner**
- volatile long long **lock\_count**
- CRITICAL\_SECTION **mutex**

## 6.571.1 Constructor & Destructor Documentation

6.571.1.1 decaf::util::concurrent::MutexHandle::MutexHandle ( ) [inline]

6.571.1.2 decaf::util::concurrent::MutexHandle::~~MutexHandle ( ) [inline]

6.571.1.3 decaf::util::concurrent::MutexHandle::MutexHandle ( ) [inline]

6.571.1.4 decaf::util::concurrent::MutexHandle::~~MutexHandle ( ) [inline]

## 6.571.2 Field Documentation

6.571.2.1 volatile long long decaf::util::concurrent::MutexHandle::lock\_count

6.571.2.2 volatile long long decaf::util::concurrent::MutexHandle::lock\_owner

6.571.2.3 CRITICAL\_SECTION decaf::util::concurrent::MutexHandle::mutex

6.571.2.4 pthread\_mutex\_t decaf::util::concurrent::MutexHandle::mutex

The documentation for this class was generated from the following files:

- src/main/decaf/internal/util/concurrent/unix/MutexHandle.h
- src/main/decaf/internal/util/concurrent/windows/MutexHandle.h

## 6.572 decaf::internal::util::concurrent::MutexImpl Class Reference

```
#include <src/main/decaf/internal/util/concurrent/MutexImpl.h>
```

## Static Public Member Functions

- static decaf::util::concurrent::MutexHandle \* **create** ()  
*Creates a Reentrant Mutex and returns the handle, throws a Runtime Exception if the Mutex cannot be created for some reason.*
- static void **destroy** (decaf::util::concurrent::MutexHandle \*handle)  
*Destroy a previously create Mutex instance.*

- static void **lock** (**decaf::util::concurrent::MutexHandle** \*handle)  
*Locks the Mutex.*
- static bool **trylock** (**decaf::util::concurrent::MutexHandle** \*handle)  
*Tries to lock the Mutex.*
- static void **unlock** (**decaf::util::concurrent::MutexHandle** \*handle)  
*Unlocks the Mutex allowing other Thread to then acquire the Lock on it.*

### 6.572.1 Member Function Documentation

**6.572.1.1 static decaf::util::concurrent::MutexHandle\*  
decaf::internal::util::concurrent::MutexImpl::create ( ) [static]**

Creates a Reentrant Mutex and returns the handle, throws a Runtime Exception if the Mutex cannot be created for some reason.

#### Returns

handle to a newly created Mutex.

**6.572.1.2 static void decaf::internal::util::concurrent::MutexImpl::destroy (  
decaf::util::concurrent::MutexHandle \* *handle* ) [static]**

Destroy a previously create Mutex instance.

#### Parameters

<i>mutex</i>	The Mutex instance to be destroyed.
--------------	-------------------------------------

**6.572.1.3 static void decaf::internal::util::concurrent::MutexImpl::lock (  
decaf::util::concurrent::MutexHandle \* *handle* ) [static]**

Locks the Mutex.

If the Mutex is already locked by another thread this method blocks until the Mutex becomes unlocked and this thread acquires the lock.

#### Parameters

<i>handle</i>	the handle to the Mutex to Lock.
---------------	----------------------------------

**6.572.1.4** `static bool decaf::internal::util::concurrent::MutexImpl::trylock ( decaf::util::concurrent::MutexHandle * handle ) [static]`

Tries to lock the Mutex.

If the Mutex is unlocked this Thread acquires the lock on the Mutex and this method returns true, if the Mutex is already locked then the lock is not acquired and this method returns false.

#### Parameters

<i>handle</i>	the handle to the Mutex to attempt to Lock.
---------------	---

#### Returns

true if the lock was acquired false otherwise.

**6.572.1.5** `static void decaf::internal::util::concurrent::MutexImpl::unlock ( decaf::util::concurrent::MutexHandle * handle ) [static]`

Unlocks the Mutex allowing other Thread to then acquire the Lock on it.

#### Parameters

<i>handle</i>	the handle to the Mutex to attempt to Lock.
---------------	---

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/MutexImpl.h

## 6.573 decaf::internal::net::Network Class Reference

Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.

```
#include <src/main/decaf/internal/net/Network.h>
```

### Public Member Functions

- virtual `~Network ()`
- `decaf::util::concurrent::Mutex * getRuntimeLock ()`  
*Gets a pointer to the **Network** (p. 2874) Runtime's Lock object, this can be used by **Network** (p. 2874) layer APIs to synchronize around certain actions such as adding a resource to the **Network** (p. 2874) layer, etc.*
- void `addNetworkResource (decaf::internal::util::Resource *value)`  
*Adds a Resource to the **Network** (p. 2874) Runtime, this resource will be held by the runtime until the Library shutdown method is called at which time all the Resources held by the **Network** (p. 2874) Runtime are destroyed.*

- `template<typename T >`  
`void addAsResource (T *value)`

### Static Public Member Functions

- `static Network * getNetworkRuntime ()`

*Gets the one and only instance of the **Network** (p. 2874) class, if this is called before the **Network** (p. 2874) layer has been initialized or after it has been shutdown then an `IllegalStateException` is thrown.*

- `static void initializeNetworking ()`

*Initialize the Networking layer.*

- `static void shutdownNetworking ()`

*Shutdown the **Network** (p. 2874) layer and free any associated resources, classes in the Decaf library that use the networking layer will now fail if used after calling the shutdown method.*

### Protected Member Functions

- `Network ()`

#### 6.573.1 Detailed Description

Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.

#### Since

1.0

## 6.573.2 Constructor & Destructor Documentation

6.573.2.1 `decaf::internal::net::Network ( )` [protected]

6.573.2.2 `virtual decaf::internal::net::Network::~~Network ( )` [virtual]

## 6.573.3 Member Function Documentation

6.573.3.1 `template<typename T > void decaf::internal::net::Network::addAsResource ( T * value )` [inline]

6.573.3.2 `void decaf::internal::net::Network::addNetworkResource ( decaf::internal::util::Resource * value )`

Adds a Resource to the **Network** (p. 2874) Runtime, this resource will be held by the runtime until the Library shutdown method is called at which time all the Resources held by the **Network** (p. 2874) Runtime are destroyed.

### Parameters

<i>value</i>	The Resource to add to the <b>Network</b> (p. 2874) Runtime.
--------------	--

### Exceptions

<i>NullPointerException</i>	if the Resource value passed is null.
-----------------------------	---------------------------------------

6.573.3.3 `static Network* decaf::internal::net::Network::getNetworkRuntime ( )` [static]

Gets the one and only instance of the **Network** (p. 2874) class, if this is called before the **Network** (p. 2874) layer has been initialized or after it has been shutdown then an `IllegalStateException` is thrown.

### Returns

pointer to the **Network** (p. 2874) runtime for the Decaf library.

6.573.3.4 `decaf::util::concurrent::Mutex* decaf::internal::net::Network::getRuntimeLock ( )`

Gets a pointer to the **Network** (p. 2874) Runtime's Lock object, this can be used by **Network** (p. 2874) layer APIs to synchronize around certain actions such as adding a resource to the **Network** (p. 2874) layer, etc.

The pointer returned is owned by the **Network** (p. 2874) runtime and should not be deleted or copied by the caller.

**Returns**

a pointer to the **Network** (p. 2874) Runtime's single Lock instance.

**6.573.3.5** `static void decaf::internal::net::Network::initializeNetworking ( ) [static]`

Initialize the Networking layer.

**6.573.3.6** `static void decaf::internal::net::Network::shutdownNetworking ( ) [static]`

Shutdown the **Network** (p. 2874) layer and free any associated resources, classes in the Decaf library that use the networking layer will now fail if used after calling the shutdown method.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**Network.h**

**6.574 activemq::commands::NetworkBridgeFilter Class Reference**

```
#include <src/main/activemq/commands/NetworkBridgeFilter.h>
```

Inheritance diagram for activemq::commands::NetworkBridgeFilter:

**Public Member Functions**

- **NetworkBridgeFilter** ( )
- virtual **~NetworkBridgeFilter** ( )
- virtual unsigned char **getDataStructureType** ( ) const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **NetworkBridgeFilter** \* **cloneDataStructure** ( ) const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** ( ) const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const



Compares the *DataStructure* (p. 1713) passed in to this one, and returns if they are equivalent.

- virtual int **getNetworkTTL** () const
- virtual void **setNetworkTTL** (int **networkTTL**)
- virtual const **Pointer**< **BrokerId** > & **getNetworkBrokerId** () const
- virtual **Pointer**< **BrokerId** > & **getNetworkBrokerId** ()
- virtual void **setNetworkBrokerId** (const **Pointer**< **BrokerId** > &**networkBrokerId**)

### Static Public Attributes

- static const unsigned char **ID\_NETWORKBRIDGEFILTER** = 91

### Protected Attributes

- int **networkTTL**
- **Pointer**< **BrokerId** > **networkBrokerId**

## 6.574.1 Constructor & Destructor Documentation

6.574.1.1 **activemq::commands::NetworkBridgeFilter::NetworkBridgeFilter** ( )

6.574.1.2 **virtual activemq::commands::NetworkBridgeFilter::~~NetworkBridgeFilter** ( )  
[virtual]

## 6.574.2 Member Function Documentation

6.574.2.1 **virtual NetworkBridgeFilter\* activemq::commands::NetworkBridgeFilter::cloneDataStructure** ( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

6.574.2.2 **virtual void activemq::commands::NetworkBridgeFilter::copyDataStructure** ( const **DataStructure** \* **src** ) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

**Parameters**

<i>src</i>	- Source Object
------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1715).

**6.574.2.3** `virtual bool activemq::commands::NetworkBridgeFilter::equals ( const DataStructure * value ) const` [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns**

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1716).

**6.574.2.4** `virtual unsigned char activemq::commands::NetworkBridgeFilter::getDataStructureType ( ) const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns**

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

- 6.574.2.5 **virtual const Pointer<BrokerId>& activemq::commands::NetworkBridgeFilter::getNetworkBrokerId ( ) const** [virtual]
- 6.574.2.6 **virtual Pointer<BrokerId>& activemq::commands::NetworkBridgeFilter::getNetworkBrokerId ( )** [virtual]
- 6.574.2.7 **virtual int activemq::commands::NetworkBridgeFilter::getNetworkTTL ( ) const** [virtual]
- 6.574.2.8 **virtual void activemq::commands::NetworkBridgeFilter::setNetworkBrokerId ( const Pointer< BrokerId > & networkBrokerId )** [virtual]
- 6.574.2.9 **virtual void activemq::commands::NetworkBridgeFilter::setNetworkTTL ( int networkTTL )** [virtual]
- 6.574.2.10 **virtual std::string activemq::commands::NetworkBridgeFilter::toString ( ) const** [virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 841).

#### 6.574.3 Field Documentation

- 6.574.3.1 **const unsigned char activemq::commands::NetworkBridgeFilter::ID\_NETWORKBRIDGEFILTER = 91** [static]
- 6.574.3.2 **Pointer<BrokerId> activemq::commands::NetworkBridgeFilter::networkBrokerId** [protected]
- 6.574.3.3 **int activemq::commands::NetworkBridgeFilter::networkTTL** [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**NetworkBridgeFilter.h**

## 6.575 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2881).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/NetworkBridgeFi
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller:

### Public Member Functions

- **NetworkBridgeFilterMarshaller ()**
- virtual **~NetworkBridgeFilterMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn)** throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

6.575

activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller  
Class Reference 2885

### 6.575.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2881).  
NOTE!: This file is auto generated - do not modify! if you need to make a change,  
please see the Java Classes in the activemq-openwire-generator module

### 6.575.2 Constructor & Destructor Documentation

6.575.2.1 **activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller**  
( ) [inline]

6.575.2.2 **virtual activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller**  
( ) [inline, virtual]

### 6.575.3 Member Function Documentation

6.575.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.575.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.575.3.3 **virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.575.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.575.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

6.575

**activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller**  
**Class Reference** **2887**  
**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.575.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.575.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/NetworkBridgeFilterMarshaller.h`

## 6.576 `activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller` Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2885).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/NetworkBridgeFi
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller`:

### Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual `~NetworkBridgeFilterMarshaller` ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )



6.576

**activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller**  
**Class Reference** 2889

---

*Write a object instance to data output stream.*

### 6.576.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2885).  
NOTE!: This file is auto generated - do not modify! if you need to make a change,  
please see the Java Classes in the activemq-openwire-generator module

### 6.576.2 Constructor & Destructor Documentation

6.576.2.1 **activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller**  
( ) [inline]

6.576.2.2 **virtual activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller**  
( ) [inline, virtual]

### 6.576.3 Member Function Documentation

6.576.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.576.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.576.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure**  
\* **dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.576.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.576.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

6.576

**activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller**  
**Class Reference** **2891**  
**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.576.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.576.3.7** `virtual void activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/NetworkBridgeFilterMarshaller.h`

## 6.577 `activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2889).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/NetworkBridgeFi
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller`:

### Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

6.577

**activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller**  
**Class Reference** **2893**

---

*Write a object instance to data output stream.*

### 6.577.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2889).  
NOTE!: This file is auto generated - do not modify! if you need to make a change,  
please see the Java Classes in the activemq-openwire-generator module

### 6.577.2 Constructor & Destructor Documentation

6.577.2.1 **activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller**  
( ) [inline]

6.577.2.2 **virtual activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller**  
( ) [inline, virtual]

### 6.577.3 Member Function Documentation

6.577.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.577.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.577.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure**  
\* **dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.577.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.577.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

6.577

**activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller**  
**Class Reference** **2895**  
**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.577.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.577.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/NetworkBridgeFilterMarshaller.h`

## 6.578 `activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller` Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2893).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFi
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller`:

### Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual `~NetworkBridgeFilterMarshaller` ()
- virtual `commands::DataStructure * createObject` () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataInputStream *dataIn`, `utils::BooleanStream *bs`) throw ( `decaf::io::IOException` )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `utils::BooleanStream *bs`) throw ( `decaf::io::IOException` )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataOutputStream *dataOut`, `utils::BooleanStream *bs`) throw ( `decaf::io::IOException` )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataInputStream *dataIn`) throw ( `decaf::io::IOException` )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataOutputStream *dataOut`) throw ( `decaf::io::IOException` )



6.578

**activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller**  
**Class Reference** 2897

---

*Write a object instance to data output stream.*

### 6.578.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2893).  
NOTE!: This file is auto generated - do not modify! if you need to make a change,  
please see the Java Classes in the activemq-openwire-generator module

### 6.578.2 Constructor & Destructor Documentation

6.578.2.1 **activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller**  
( ) [inline]

6.578.2.2 **virtual activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller**  
( ) [inline, virtual]

### 6.578.3 Member Function Documentation

6.578.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.578.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.578.3.3 **virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure**  
\* **dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw (  
**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.578.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.578.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

6.578

**activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller**  
**Class Reference** **2899**  
**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.578.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.578.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h`

## 6.579 `activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2897).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFi
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller`:

### Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual `~NetworkBridgeFilterMarshaller` ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

6.579

**activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller**  
**Class Reference** **2901**

---

*Write a object instance to data output stream.*

### 6.579.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2897).  
NOTE!: This file is auto generated - do not modify! if you need to make a change,  
please see the Java Classes in the activemq-openwire-generator module

### 6.579.2 Constructor & Destructor Documentation

6.579.2.1 **activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller**  
( ) [inline]

6.579.2.2 **virtual activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller**  
( ) [inline, virtual]

### 6.579.3 Member Function Documentation

6.579.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.579.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.579.3.3 **virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure**  
\* **dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw (  
**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.579.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.579.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

6.579

**activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller**  
**Class Reference** **2903**  
**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.579.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.579.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h`

## 6.580 `activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller` Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2901).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFi
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller`:

### Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )



6.580

**activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller**  
**Class Reference** **2905**

---

*Write a object instance to data output stream.*

### 6.580.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2901).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.580.2 Constructor & Destructor Documentation

6.580.2.1 **activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller**  
( ) [inline]

6.580.2.2 **virtual activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller**  
( ) [inline, virtual]

### 6.580.3 Member Function Documentation

6.580.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.580.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.580.3.3 **virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.580.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.580.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

6.580

**activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller**  
**Class Reference** **2907**  
**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.580.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.580.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFilterMarshaller.h`

## 6.581 decaf::net::NoRouteToHostException Class Reference

```
#include <src/main/decaf/net/NoRouteToHostException.h>
```

Inheritance diagram for `decaf::net::NoRouteToHostException`:

### Public Member Functions

- **NoRouteToHostException** () throw ()  
*Default Constructor.*
- **NoRouteToHostException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **NoRouteToHostException** (const **NoRouteToHostException** &ex) throw ()  
*Copy Constructor.*
- **NoRouteToHostException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **NoRouteToHostException** (const std::exception \*cause) throw ()  
*Constructor.*
- **NoRouteToHostException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **NoRouteToHostException** \* clone () const  
*Clones this exception.*
- virtual ~**NoRouteToHostException** () throw ()

### 6.581.1 Constructor & Destructor Documentation

6.581.1.1 `decaf::net::NoRouteToHostException::NoRouteToHostException ( ) throw ()`  
[inline]

Default Constructor.

**6.581.1.2** `decaf::net::NoRouteToHostException::NoRouteToHostException ( const Exception & ex ) throw () [inline]`

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.581.1.3** `decaf::net::NoRouteToHostException::NoRouteToHostException ( const NoRouteToHostException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.581.1.4** `decaf::net::NoRouteToHostException::NoRouteToHostException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.581.1.5** `decaf::net::NoRouteToHostException::NoRouteToHostException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.581.1.6** `decaf::net::NoRouteToHostException::NoRouteToHostException ( const char * file,  
const int lineNumber, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.  
Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.581.1.7** `virtual decaf::net::NoRouteToHostException::~~NoRouteToHostException ( ) throw ()`  
`[inline, virtual]`

#### 6.581.2 Member Function Documentation

**6.581.2.1** `virtual NoRouteToHostException* decaf::net::NoRouteToHostException::clone ( ) const` `[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 3629).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/NoRouteToHostException.h`

### 6.582 decaf::security::NoSuchAlgorithmException Class Reference

```
#include <src/main/decaf/security/NoSuchAlgorithmException.h>
```

Inheritance diagram for `decaf::security::NoSuchAlgorithmException`:

#### Public Member Functions

- `NoSuchAlgorithmException () throw ()`

*Default Constructor.*

- **NoSuchAlgorithmException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **NoSuchAlgorithmException** (const NoSuchAlgorithmException &ex) throw ()  
*Copy Constructor.*
- **NoSuchAlgorithmException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **NoSuchAlgorithmException** (const std::exception \*cause) throw ()  
*Constructor.*
- **NoSuchAlgorithmException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **NoSuchAlgorithmException \* clone** () const  
*Clones this exception.*
- virtual **~NoSuchAlgorithmException** () throw ()

## 6.582.1 Constructor & Destructor Documentation

**6.582.1.1** decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException ( ) throw ()  
[inline]

Default Constructor.

**6.582.1.2** decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException ( const Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.582.1.3** decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException ( const NoSuchAlgorithmException & ex ) throw () [inline]

Copy Constructor.

**Parameters**

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.582.1.4** `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.  
Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.582.1.5** `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException ( const std::exception * cause ) throw () [inline]`

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.582.1.6** `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.  
Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
...	list of primitives that are formatted into the message



6.582.1.7 `virtual decaf::security::NoSuchAlgorithmException::~~NoSuchAlgorithmException ( )  
throw () [inline, virtual]`

## 6.582.2 Member Function Documentation

6.582.2.1 `virtual NoSuchElementException* decaf::security::NoSuchAlgorithmException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

A deep copy of this exception.

Reimplemented from `decaf::security::GeneralSecurityException` (p. 2037).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/NoSuchAlgorithmException.h`

## 6.583 `decaf::lang::exceptions::NoSuchElementException` Class Reference

```
#include <src/main/decaf/lang/exceptions/NoSuchElementException.h>
```

Inheritance diagram for `decaf::lang::exceptions::NoSuchElementException`:

### Public Member Functions

- **`NoSuchElementException () throw ()`**  
*Default Constructor.*
- **`NoSuchElementException (const Exception &ex) throw ()`**  
*Conversion Constructor from some other `Exception` (p. 1886).*
- **`NoSuchElementException (const NoSuchElementException &ex) throw ()`**  
*Copy Constructor.*
- **`NoSuchElementException (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()`**  
*Constructor - Initializes the file name and line number where this message occurred.*

- **NoSuchElementException** (const std::exception \***cause**) throw ()  
*Constructor.*
- **NoSuchElementException** (const char \***file**, const int **lineNumber**, const char \***msg**,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **NoSuchElementException** \* **clone** () const  
*Clones this exception.*
- virtual ~**NoSuchElementException** () throw ()

### 6.583.1 Constructor & Destructor Documentation

6.583.1.1 **decaf::lang::exceptions::NoSuchElementException::NoSuchElementException ( )**  
**throw ()** [inline]

Default Constructor.

6.583.1.2 **decaf::lang::exceptions::NoSuchElementException::NoSuchElementException ( const Exception & **ex** )** throw () [inline]

Conversion Constructor from some other **Exception** (p. 1886).

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

6.583.1.3 **decaf::lang::exceptions::NoSuchElementException::NoSuchElementException ( const NoSuchElementException & **ex** )** throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

6.583.1.4 **decaf::lang::exceptions::NoSuchElementException::NoSuchElementException ( const char \* **file**, const int **lineNumber**, const std::exception \* **cause**, const char \* **msg**, ... )** throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

## 6.583 decaf::lang::exceptions::NoSuchElementException Class Reference 2915

### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.583.1.5** decaf::lang::exceptions::NoSuchElementException::NoSuchElementException ( const std::exception \* *cause* ) throw () [inline]

Constructor.

### Parameters

<i>cause</i>	<b>Pointer</b> (p.3032) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.583.1.6** decaf::lang::exceptions::NoSuchElementException::NoSuchElementException ( const char \* *file*, const int *lineNumber*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.583.1.7** virtual decaf::lang::exceptions::NoSuchElementException::~~NoSuchElementException ( ) throw () [inline, virtual]

## 6.583.2 Member Function Documentation

**6.583.2.1** virtual NoSuchElementException\* decaf::lang::exceptions::NoSuchElementException::clone ( ) const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

an new **Exception** (p.1886) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1889).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/NoSuchElementException.h`

## 6.584 decaf::security::NoSuchProviderException Class Reference

```
#include <src/main/decaf/security/NoSuchProviderException.h>
```

Inheritance diagram for `decaf::security::NoSuchProviderException`:

### Public Member Functions

- **NoSuchProviderException** () throw ()  
*Default Constructor.*
- **NoSuchProviderException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **NoSuchProviderException** (const NoSuchProviderException &ex) throw ()  
*Copy Constructor.*
- **NoSuchProviderException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **NoSuchProviderException** (const std::exception \*cause) throw ()  
*Constructor.*
- **NoSuchProviderException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **NoSuchProviderException** \* clone () const  
*Clones this exception.*
- virtual ~**NoSuchProviderException** () throw ()

### 6.584.1 Constructor & Destructor Documentation

6.584.1.1 **decaf::security::NoSuchProviderException::NoSuchProviderException** ( ) throw ()  
[inline]

Default Constructor.

**6.584.1.2** `decaf::security::NoSuchProviderException::NoSuchProviderException ( const Exception & ex ) throw () [inline]`

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.584.1.3** `decaf::security::NoSuchProviderException::NoSuchProviderException ( const NoSuchProviderException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.584.1.4** `decaf::security::NoSuchProviderException::NoSuchProviderException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.584.1.5** `decaf::security::NoSuchProviderException::NoSuchProviderException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.584.1.6** `decaf::security::NoSuchProviderException::NoSuchProviderException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
<i>...</i>	list of primitives that are formatted into the message

**6.584.1.7** `virtual decaf::security::NoSuchProviderException::~~NoSuchProviderException ( ) throw () [inline, virtual]`

#### 6.584.2 Member Function Documentation

**6.584.2.1** `virtual NoSuchProviderException* decaf::security::NoSuchProviderException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

A deep copy of this exception.

Reimplemented from `decaf::security::GeneralSecurityException` (p. 2037).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/NoSuchProviderException.h`

### 6.585 decaf::lang::exceptions::NullPointerException Class Reference

```
#include <src/main/decaf/lang/exceptions/NullPointerException.h>
```

Inheritance diagram for `decaf::lang::exceptions::NullPointerException`:

#### Public Member Functions

- `NullPointerException () throw ()`

*Default Constructor.*

- **NullPointerException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1886).*
- **NullPointerException** (const **NullPointerException** &ex) throw ()  
*Copy Constructor.*
- **NullPointerException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **NullPointerException** (const std::exception \*cause) throw ()  
*Constructor.*
- **NullPointerException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **NullPointerException** \* clone () const  
*Clones this exception.*
- virtual ~**NullPointerException** () throw ()

## 6.585.1 Constructor & Destructor Documentation

**6.585.1.1** decaf::lang::exceptions::NullPointerException::NullPointerException ( ) throw ()  
 [inline]

Default Constructor.

**6.585.1.2** decaf::lang::exceptions::NullPointerException::NullPointerException ( const **Exception** & ex ) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1886).

### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

**6.585.1.3** decaf::lang::exceptions::NullPointerException::NullPointerException ( const **NullPointerException** & ex ) throw () [inline]

Copy Constructor.

**Parameters**

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

**6.585.1.4** `decaf::lang::exceptions::NullPointerException::NullPointerException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.  
Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.585.1.5** `decaf::lang::exceptions::NullPointerException::NullPointerException ( const std::exception * cause ) throw () [inline]`

Constructor.

**Parameters**

<i>cause</i>	<b>Pointer</b> (p. 3032) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

**6.585.1.6** `decaf::lang::exceptions::NullPointerException::NullPointerException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.  
Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message



6.585.1.7 virtual decaf::lang::exceptions::NullPointerException::~~NullPointerException ( )  
throw () [inline, virtual]

## 6.585.2 Member Function Documentation

6.585.2.1 virtual NullPointerException\* decaf::lang::exceptions::NullPointerException::clone ( ) const  
[inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

an new **Exception** (p. 1886) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1889).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/NullPointerException.h

## 6.586 decaf::lang::Number Class Reference

The abstract class **Number** (p. 2918) is the superclass of classes **Byte** (p. 969), **Double** (p. 1841), **Float** (p. 1961), **Integer** (p. 2143), **Long** (p. 2495), and **Short** (p. 3538).

```
#include <src/main/decaf/lang/Number.h>
```

Inheritance diagram for decaf::lang::Number:

### Public Member Functions

- virtual ~**Number** ()
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual double **doubleValue** () const =0  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const =0  
*Answers the float value which the receiver represents.*
- virtual int **intValue** () const =0

*Answers the int value which the receiver represents.*

- virtual long long **longValue** () const =0

*Answers the long value which the receiver represents.*

- virtual short **shortValue** () const

*Answers the short value which the receiver represents.*

### 6.586.1 Detailed Description

The abstract class **Number** (p. 2918) is the superclass of classes **Byte** (p. 969), **Double** (p. 1841), **Float** (p. 1961), **Integer** (p. 2143), **Long** (p. 2495), and **Short** (p. 3538). Subclasses of **Number** (p. 2918) must provide methods to convert the represented numeric value to byte, double, float, int, long, and short.

### 6.586.2 Constructor & Destructor Documentation

6.586.2.1 virtual decaf::lang::Number::~~Number ( ) [inline, virtual]

### 6.586.3 Member Function Documentation

6.586.3.1 virtual unsigned char decaf::lang::Number::byteValue ( ) const [inline, virtual]

Answers the byte value which the receiver represents.

#### Returns

byte the value of the receiver.

Reimplemented in **decaf::lang::Byte** (p. 972), **decaf::lang::Character** (p. 1129), **decaf::lang::Double** (p. 1844), **decaf::lang::Float** (p. 1964), **decaf::lang::Integer** (p. 2147), **decaf::lang::Long** (p. 2499), and **decaf::lang::Short** (p. 3541).

6.586.3.2 virtual double decaf::lang::Number::doubleValue ( ) const [pure virtual]

Answers the double value which the receiver represents.

#### Returns

double the value of the receiver.

Implemented in **decaf::lang::Byte** (p. 973), **decaf::lang::Character** (p. 1130), **decaf::lang::Double** (p. 1846), **decaf::lang::Float** (p. 1966), **decaf::lang::Integer** (p. 2148), **decaf::lang::Long** (p. 2500), **decaf::lang::Short** (p. 3542), and **decaf::util::concurrent::atomic::AtomicInteger** (p. 751).

**6.586.3.3** virtual float decaf::lang::Number::floatValue ( ) const [pure virtual]

Answers the float value which the receiver represents.

#### Returns

float the value of the receiver.

Implemented in **decaf::lang::Byte** (p. 974), **decaf::lang::Character** (p. 1131), **decaf::lang::Double** (p. 1847), **decaf::lang::Float** (p. 1967), **decaf::lang::Integer** (p. 2149), **decaf::lang::Long** (p. 2501), **decaf::lang::Short** (p. 3543), and **decaf::util::concurrent::atomic::AtomicInteger** (p. 751).

**6.586.3.4** virtual int decaf::lang::Number::intValue ( ) const [pure virtual]

Answers the int value which the receiver represents.

#### Returns

int the value of the receiver.

Implemented in **decaf::lang::Byte** (p. 974), **decaf::lang::Character** (p. 1131), **decaf::lang::Double** (p. 1847), **decaf::lang::Float** (p. 1968), **decaf::lang::Integer** (p. 2150), **decaf::lang::Long** (p. 2501), **decaf::lang::Short** (p. 3543), and **decaf::util::concurrent::atomic::AtomicInteger** (p. 753).

**6.586.3.5** virtual long long decaf::lang::Number::longValue ( ) const [pure virtual]

Answers the long value which the receiver represents.

#### Returns

long long the value of the receiver.

Implemented in **decaf::lang::Byte** (p. 974), **decaf::lang::Character** (p. 1132), **decaf::lang::Double** (p. 1849), **decaf::lang::Float** (p. 1969), **decaf::lang::Integer** (p. 2150), **decaf::lang::Long** (p. 2502), **decaf::lang::Short** (p. 3543), and **decaf::util::concurrent::atomic::AtomicInteger** (p. 753).

**6.586.3.6** virtual short decaf::lang::Number::shortValue ( ) const [inline, virtual]

Answers the short value which the receiver represents.

#### Returns

short the value of the receiver.

Reimplemented in **decaf::lang::Byte** (p. 977), **decaf::lang::Character** (p. 1134), **decaf::lang::Double** (p. 1850), **decaf::lang::Float** (p. 1971), **decaf::lang::Integer** (p. 2155), **decaf::lang::Long** (p. 2507), and **decaf::lang::Short** (p. 3546).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Number.h`

## 6.587 decaf::lang::exceptions::NumberFormatException Class Reference

```
#include <src/main/decaf/lang/exceptions/NumberFormatException.h>
```

Inheritance diagram for `decaf::lang::exceptions::NumberFormatException`:

### Public Member Functions

- **NumberFormatException** ()  
*Default Constructor.*
- **NumberFormatException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1886).*
- **NumberFormatException** (const **NumberFormatException** &ex) throw ()  
*Copy Constructor.*
- **NumberFormatException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **NumberFormatException** (const std::exception \*cause) throw ()  
*Constructor.*
- **NumberFormatException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **NumberFormatException** \* clone () const  
*Clones this exception.*
- virtual ~**NumberFormatException** () throw ()

### 6.587.1 Constructor & Destructor Documentation

**6.587.1.1** `decaf::lang::exceptions::NumberFormatException::NumberFormatException ( )`  
`[inline]`

Default Constructor.

Referenced by clone().

**6.587.1.2** `decaf::lang::exceptions::NumberFormatException::NumberFormatException ( const Exception & ex ) throw ()` `[inline]`

Conversion Constructor from some other **Exception** (p. 1886).

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

**6.587.1.3** `decaf::lang::exceptions::NumberFormatException::NumberFormatException ( const NumberFormatException & ex ) throw ()` `[inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

**6.587.1.4** `decaf::lang::exceptions::NumberFormatException::NumberFormatException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

References decaf::lang::Exception::buildMessage(), and decaf::lang::Exception::setMark().

**6.587.1.5** `decaf::lang::exceptions::NumberFormatException::NumberFormatException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	<b>Pointer</b> (p. 3032) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

**6.587.1.6** `decaf::lang::exceptions::NumberFormatException::NumberFormatException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

References `decaf::lang::Exception::buildMessage()`, and `decaf::lang::Exception::setMark()`.

**6.587.1.7** `virtual decaf::lang::exceptions::NumberFormatException::~~NumberFormatException ( ) throw () [inline, virtual]`

## 6.587.2 Member Function Documentation

**6.587.2.1** `virtual NumberFormatException* decaf::lang::exceptions::NumberFormatException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

an new **Exception** (p. 1886) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1889).

References `NumberFormatException()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/NumberFormatException.h`

## 6.588 cms::ObjectMessage Class Reference

Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.

```
#include <src/main/cms/ObjectMessage.h>
```

Inheritance diagram for cms::ObjectMessage:

### Public Member Functions

- virtual ~ObjectMessage ()

#### 6.588.1 Detailed Description

Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object. serialized **ObjectMessage** (p.2924) s.

Since

1.0

#### 6.588.2 Constructor & Destructor Documentation

6.588.2.1 virtual cms::ObjectMessage::~ObjectMessage ( ) [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/cms/ObjectMessage.h

## 6.589 decaf::internal::net::ssl::openssl::OpenSSLContextSpi Class Reference

Provides an SSLContext that wraps the OpenSSL API.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLContextSpi:

### Public Member Functions

- OpenSSLContextSpi ()

- virtual `~OpenSSLContextSpi ()`
- virtual void **providerInit** (`security::SecureRandom *random`)

*Perform the initialization of this Context.*

**Parameters**

random	Pointer to an instance of a secure random number generator.
--------	---

**Exceptions**

NullPointerException	if the <i>SecureRandom</i> instance is <i>NULL</i> .
KeyManagementException	if an error occurs while initializing the context.

- virtual **decaf::net::SocketFactory \* providerGetSocketFactory ()**

*Returns a **SocketFactory** (p. 3629) instance that can be used to create new **SSLSocket** (p. 3670) objects.*

*The **SocketFactory** (p. 3629) is owned by the Service Provider and should not be destroyed by the caller.*

**Returns**

***SocketFactory** (p. 3629) instance that can be used to create new **SSL**Sockets.*

**Exceptions**

IllegalStateException	if the <i>SSLContextSpi</i> (p. 3655) object requires initialization but has not been initialized yet.
-----------------------	--

- virtual **decaf::net::ServerSocketFactory \* providerGetServerSocketFactory ()**

*Returns a **ServerSocketFactory** (p. 3457) instance that can be used to create new **SSLServerSocket** (p. 3662) objects.*

*The **ServerSocketFactory** (p. 3457) is owned by the Service Provider and should not be destroyed by the caller.*

**Returns**

***SocketFactory** (p. 3629) instance that can be used to create new **SSLServerSockets**.*

**Exceptions**

IllegalStateException	if the <i>SSLContextSpi</i> (p. 3655) object requires initialization but has not been initialized yet.
-----------------------	--

## Friends

- class **OpenSSLSocket**
- class **OpenSSLSocketFactory**

### 6.589.1 Detailed Description

Provides an **SSLContext** that wraps the **OpenSSL** API.



## 6.589 decaf::internal::net::ssl::openssl::OpenSSLContextSpi Class Reference

Since

1.0

### 6.589.2 Constructor & Destructor Documentation

6.589.2.1 `decaf::internal::net::ssl::openssl::OpenSSLContextSpi::OpenSSLContextSpi ( )`

6.589.2.2 `virtual decaf::internal::net::ssl::openssl::OpenSSLContextSpi::~~OpenSSLContextSpi ( ) [virtual]`

### 6.589.3 Member Function Documentation

6.589.3.1 `virtual decaf::net::ServerSocketFactory* decaf::internal::net::ssl::openssl::OpenSSLContextSpi::providerGetServerSocketFactory ( ) [virtual]`

Returns a **ServerSocketFactory** (p. 3457) instance that can be used to create new **SSLServerSocket** (p. 3662) objects.

The **ServerSocketFactory** (p. 3457) is owned by the Service Provider and should not be destroyed by the caller.

#### Returns

**SocketFactory** (p. 3629) instance that can be used to create new **SSLServerSockets**.

#### Exceptions

<i>IllegalStateException</i>	if the <b>SSLContextSpi</b> (p. 3655) object requires initialization but has not been initialized yet.
------------------------------	--

Implements **decaf::net::ssl::SSLContextSpi** (p. 3657).

6.589.3.2 `virtual decaf::net::SocketFactory* decaf::internal::net::ssl::openssl::OpenSSLContextSpi::providerGetSocketFactory ( ) [virtual]`

Returns a **SocketFactory** (p. 3629) instance that can be used to create new **SSLSocket** (p. 3670) objects.

The **SocketFactory** (p. 3629) is owned by the Service Provider and should not be destroyed by the caller.

#### Returns

**SocketFactory** (p. 3629) instance that can be used to create new **SSLSockets**.

#### Exceptions

<i>IllegalStateException</i>	if the <b>SSLContextSpi</b> (p. 3655) object requires initialization but has not been initialized yet.
------------------------------	--

Implements **decaf::net::ssl::SSLContextSpi** (p. 3657).

**6.589.3.3** `virtual void decaf::internal::net::ssl::openssl::OpenSSLContextSpi::providerInit ( security::SecureRandom * random )` [virtual]

Perform the initialization of this Context.

#### Parameters

<i>random</i>	Pointer to an instance of a secure random number generator.
---------------	---

#### Exceptions

<i>NullPointerException</i>	if the SecureRandom instance is NULL.
<i>KeyManagementException</i>	if an error occurs while initializing the context.

Implements **decaf::net::ssl::SSLContextSpi** (p. 3658).

## 6.589.4 Friends And Related Function Documentation

**6.589.4.1** `friend class OpenSSLSocket` [friend]

**6.589.4.2** `friend class OpenSSLSocketFactory` [friend]

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h`

## 6.590 decaf::internal::net::ssl::openssl::OpenSSLParameters Class Reference

Container class for parameters that are Common to OpenSSL socket classes.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h>
```

#### Public Member Functions

- `virtual ~OpenSSLParameters ()`
- `bool getNeedClientAuth () const`
- `void setNeedClientAuth (bool value)`

- bool **getWantClientAuth** () const
- void **setWantClientAuth** (bool value)
- bool **getUseClientMode** () const
- void **setUseClientMode** (bool value)
- std::vector< std::string > **getSupportedCipherSuites** () const
- std::vector< std::string > **getSupportedProtocols** () const
- std::vector< std::string > **getEnabledCipherSuites** () const
- void **setEnabledCipherSuites** (const std::vector< std::string > &suites)
- std::vector< std::string > **getEnabledProtocols** () const
- void **setEnabledProtocols** (const std::vector< std::string > &protocols)
- **OpenSSLParameters \* clone** () const

*Creates a clone of this object such that all settings are transferred to a new instance of an SSL object whose parent is the same SSL\_CTX as this object's.*

### 6.590.1 Detailed Description

Container class for parameters that are Common to OpenSSL socket classes.

Since

1.0

### 6.590.2 Constructor & Destructor Documentation

6.590.2.1 virtual decaf::internal::net::ssl::openssl::OpenSSLParameters::~~OpenSSLParameters  
( ) [virtual]

### 6.590.3 Member Function Documentation

6.590.3.1 **OpenSSLParameters\*** decaf::internal::net::ssl::openssl::OpenSSLParameters::clone  
( ) const

Creates a clone of this object such that all settings are transferred to a new instance of an SSL object whose parent is the same SSL\_CTX as this object's.

- 6.590.3.2 `std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getEnabledCipherSuites ( ) const`
- 6.590.3.3 `std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getEnabledProtocols ( ) const`
- 6.590.3.4 `bool decaf::internal::net::ssl::openssl::OpenSSLParameters::getNeedClientAuth ( ) const [inline]`
- 6.590.3.5 `std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getSupportedCipherSuites ( ) const`
- 6.590.3.6 `std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getSupportedProtocols ( ) const`
- 6.590.3.7 `bool decaf::internal::net::ssl::openssl::OpenSSLParameters::getUseClientMode ( ) const [inline]`
- 6.590.3.8 `bool decaf::internal::net::ssl::openssl::OpenSSLParameters::getWantClientAuth ( ) const [inline]`
- 6.590.3.9 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setEnabledCipherSuites ( const std::vector< std::string > & suites )`
- 6.590.3.10 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setEnabledProtocols ( const std::vector< std::string > & protocols )`
- 6.590.3.11 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setNeedClientAuth ( bool value ) [inline]`
- 6.590.3.12 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setUseClientMode ( bool value ) [inline]`
- 6.590.3.13 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setWantClientAuth ( bool value ) [inline]`

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h`

## 6.591 decaf::internal::net::ssl::openssl::OpenSSLServerSocket Class Reference

SSLServerSocket based on OpenSSL library code.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLServerSocket:

## Public Member Functions

- **OpenSSLServerSocket** (**OpenSSLParameters** \*parameters)
- virtual **~OpenSSLServerSocket** ()
- virtual std::vector< std::string > **getSupportedCipherSuites** () const  
*Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 3662).*  
*Normally not all of these cipher suites will be enabled on the **Socket** (p. 3607).*

### Returns

*a vector containing the names of all the supported cipher suites.*

- virtual std::vector< std::string > **getSupportedProtocols** () const  
*Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 3662) instance.*

### Returns

*a vector containing the names of all the supported protocols.*

- virtual std::vector< std::string > **getEnabledCipherSuites** () const  
*Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 3662).*

### Returns

*vector of the names of all enabled Cipher Suites.*

- virtual void **setEnabledCipherSuites** (const std::vector< std::string > &suites)

*Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 3662) connection.*

*Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.*

### Parameters

suites	An Vector of names for all the Cipher Suites that are to be enabled.
--------	--

### Exceptions

IllegalArgumentExcep- tion	if the vector is empty or one of the names is invalid.
-------------------------------	--

- virtual std::vector< std::string > **getEnabledProtocols** () const  
*Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 3662).*

### Returns

*vector of the names of all enabled Protocols.*

- virtual void **setEnabledProtocols** (const std::vector< std::string > &protocols)

*Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 3662) connection.*

*Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.*

**Parameters**

protocols	An Vector of names for all the Protocols that are to be enabled.
-----------	--

**Exceptions**

IllegalArgumentExcep- tion	if the vector is empty or one of the names is invalid.
-------------------------------	--

- virtual bool **getWantClientAuth** () const

**Returns**

*true if the **Socket** (p. 3607) request client Authentication.*

- virtual void **setWantClientAuth** (bool value)

*Sets whether or not this **Socket** (p. 3607) will request Client Authentication.*

*If set to true the **Socket** (p. 3607) (when used in server mode) will request that the client authenticate itself, if the client doesn't send authentication the socket will still allow negotiation to continue.*

**Parameters**

value	Whether the server socket should request client authentication.
-------	---

- virtual bool **getNeedClientAuth** () const

**Returns**

*true if the **Socket** (p. 3607) requires client Authentication.*

- virtual void **setNeedClientAuth** (bool value)

*Sets whether or not this **Socket** (p. 3607) will require Client Authentication.*

*If set to true the **Socket** (p. 3607) (when used in server mode) will require that the client authenticate itself, if the client doesn't send authentication the socket will not allow negotiation to continue.*

**Parameters**

value	Whether the server socket should require client authentication.
-------	---

- virtual **decaf::net::Socket** \* **accept** () throw ( decaf::io::IOException )

*Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 3447), the caller blocks until a connection is made.*

*If the **SO\_TIMEOUT** option is set this method could throw a **SocketTimeoutException** (p. 3650) if the operation times out.*

**Returns**

*a new **Socket** (p. 3607) object pointer. Never returns NULL, the returned pointer*

## 6.591 decaf::internal::net::ssl::openssl::OpenSSLServerSocket Class Reference 2035

*is owned by the caller and must be explicitly freed by them.*

### Exceptions

<b>IOException</b>	<i>if an I/O error occurs while binding the socket.</i>
<b>SocketException</b> (p. 3627)	<i>if an error occurs while blocking on the accept call.</i>
<b>SocketTimeoutException</b> (p. 3650)	<i>if the SO_TIMEOUT option was used and the accept timed out.</i>

### 6.591.1 Detailed Description

SSLServerSocket based on OpenSSL library code.

Since

1.0

### 6.591.2 Constructor & Destructor Documentation

6.591.2.1 **decaf::internal::net::ssl::openssl::OpenSSLServerSocket::OpenSSLServerSocket ( OpenSSLParameters \* parameters )**

6.591.2.2 **virtual decaf::internal::net::ssl::openssl::OpenSSLServerSocket::~~OpenSSLServerSocket ( ) [virtual]**

### 6.591.3 Member Function Documentation

6.591.3.1 **virtual decaf::net::Socket\* decaf::internal::net::ssl::openssl::OpenSSLServerSocket::accept ( ) throw ( decaf::io::IOException ) [virtual]**

Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 3447), the caller blocks until a connection is made.

If the SO\_TIMEOUT option is set this method could throw a **SocketTimeoutException** (p. 3650) if the operation times out.

### Returns

a new **Socket** (p. 3607) object pointer. Never returns NULL, the returned pointer is owned by the caller and must be explicitly freed by them.

### Exceptions

<i>IOException</i>	<i>if an I/O error occurs while binding the socket.</i>
<b>SocketException</b> (p. 3627)	<i>if an error occurs while blocking on the accept call.</i>
<b>SocketTimeoutException</b> (p. 3650)	<i>if the SO_TIMEOUT option was used and the accept timed out.</i>

Reimplemented from **decaf::net::ServerSocket** (p. 3451).

```
6.591.3.2  virtual std::vector<std::string> de-  
           caf::internal::net::ssl::openssl::OpenSSLServerSocket::getEnabledCipherSuites ( )  
           const [virtual]
```

Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 3662).

#### Returns

vector of the names of all enabled Cipher Suites.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3665).

```
6.591.3.3  virtual std::vector<std::string> de-  
           caf::internal::net::ssl::openssl::OpenSSLServerSocket::getEnabledProtocols ( )  
           const [virtual]
```

Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 3662).

#### Returns

vector of the names of all enabled Protocols.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3665).

```
6.591.3.4  virtual bool decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getNeedClientAuth  
           ( ) const [virtual]
```

#### Returns

true if the **Socket** (p. 3607) requires client Authentication.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3665).

```
6.591.3.5  virtual std::vector<std::string> de-  
           caf::internal::net::ssl::openssl::OpenSSLServerSocket::getSupportedCipherSuites ( )  
           const [virtual]
```

Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 3662).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 3607).

#### Returns

a vector containing the names of all the supported cipher suites.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3666).



## 6.591 `decaf::internal::net::ssl::openssl::OpenSSLServerSocket` Class Reference 2937

**6.591.3.6** `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getSupportedProtocols ( ) const [virtual]`

Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 3662) instance.

### Returns

a vector containing the names of all the supported protocols.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3666).

**6.591.3.7** `virtual bool decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getWantClientAuth ( ) const [virtual]`

### Returns

true if the **Socket** (p. 3607) request client Authentication.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3666).

**6.591.3.8** `virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setEnabledCipherSuites ( const std::vector< std::string > & suites ) [virtual]`

Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 3662) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

### Parameters

<i>suites</i>	An Vector of names for all the Cipher Suites that are to be enabled.
---------------	--

### Exceptions

<i>IllegalArgumentException</i>	if the vector is empty or one of the names is invalid.
---------------------------------	--

Implements **decaf::net::ssl::SSLServerSocket** (p. 3666).

**6.591.3.9** `virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setEnabledProtocols ( const std::vector< std::string > & protocols ) [virtual]`

Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 3662) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

**Parameters**

<i>protocols</i>	An Vector of names for all the Protocols that are to be enabled.
------------------	--

**Exceptions**

<i>IllegalArgumentEx-ception</i>	if the vector is empty or one of the names is invalid.
----------------------------------	--

Implements **decaf::net::ssl::SSLServerSocket** (p. 3667).

**6.591.3.10** `virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setNeedClientAuth ( bool value ) [virtual]`

Sets whether or not this **Socket** (p. 3607) will require Client Authentication.

If set to true the **Socket** (p. 3607) (when used in server mode) will require that the client authenticate itself, if the client doesn't send authentication the socket will not allow negotiation to continue.

**Parameters**

<i>value</i>	Whether the server socket should require client authentication.
--------------	---

Implements **decaf::net::ssl::SSLServerSocket** (p. 3667).

**6.591.3.11** `virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setWantClientAuth ( bool value ) [virtual]`

Sets whether or not this **Socket** (p. 3607) will request Client Authentication.

If set to true the **Socket** (p. 3607) (when used in server mode) will request that the client authenticate itself, if the client doesn't send authentication the socket will still allow negotiation to continue.

**Parameters**

<i>value</i>	Whether the server socket should request client authentication.
--------------	---

Implements **decaf::net::ssl::SSLServerSocket** (p. 3667).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLServerSocket.h**

## 6.592 decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory Class Reference

SSLServerSocketFactory that creates Server Sockets that use OpenSSL.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory:

## Public Member Functions

- **OpenSSLServerSocketFactory** (**OpenSSLContextSpi** \*parent)
- virtual **~OpenSSLServerSocketFactory** ()
- virtual **decaf::net::ServerSocket** \* **createServerSocket** ()

Create a new **ServerSocket** (p. 3447) that is unbound.  
The **ServerSocket** (p. 3447) will have been configured with the defaults from the factory.

### Returns

new **ServerSocket** (p. 3447) pointer that is owned by the caller.

### Exceptions

IOException	if the <b>ServerSocket</b> (p. 3447) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket** \* **createServerSocket** (int port)

Create a new **ServerSocket** (p. 3447) that is bound to the given port.  
The **ServerSocket** (p. 3447) will have been configured with the defaults from the factory.

### Parameters

port	The port to bind the <b>ServerSocket</b> (p. 3447) to.
------	--

### Returns

new **ServerSocket** (p. 3447) pointer that is owned by the caller.

### Exceptions

IOException	if the <b>ServerSocket</b> (p. 3447) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket** \* **createServerSocket** (int port, int backlog)

Create a new **ServerSocket** (p. 3447) that is bound to the given port.  
The **ServerSocket** (p. 3447) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3447) will use the specified connection backlog setting.

### Parameters

port	The port to bind the <b>ServerSocket</b> (p. 3447) to.
backlog	The number of pending connect request the <b>ServerSocket</b> (p. 3447) can queue.

### Returns

new **ServerSocket** (p. 3447) pointer that is owned by the caller.

### Exceptions

IOException	if the <b>ServerSocket</b> (p. 3447) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket \* createServerSocket** (int port, int backlog, const **decaf::net::InetAddress \*address**)

Create a new **ServerSocket** (p. 3447) that is bound to the given port. The **ServerSocket** (p. 3447) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3447) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 3447) will listen on all interfaces.

**Parameters**

port	The port to bind the <b>ServerSocket</b> (p. 3447) to.
backlog	The number of pending connect request the <b>ServerSocket</b> (p. 3447) can queue.
address	The address of the interface on the local machine to bind to.

**Returns**

new **ServerSocket** (p. 3447) pointer that is owned by the caller.

**Exceptions**

IOException	if the <b>ServerSocket</b> (p. 3447) cannot be created for some reason.
-------------	---

- virtual std::vector< std::string > **getDefaultCipherSuites** ()

Returns the list of cipher suites which are enabled by default. Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

**Returns**

an STL vector containing the list of cipher suites enabled by default.

**See also**

**getSupportedCipherSuites**() (p. 3670)

- virtual std::vector< std::string > **getSupportedCipherSuites** ()

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

**Returns**

an STL vector containing the list of supported cipher suites.

**See also**

**getDefaultCipherSuites**() (p. 3669)

### 6.592.1 Detailed Description

SSLServerSocketFactory that creates Server Sockets that use OpenSSL.

**Since**

1.0

## 6.592.2 Constructor & Destructor Documentation

6.592.2.1 `decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::OpenSSLServerSocketFactory ( OpenSSLContextSpi * parent )`

6.592.2.2 `virtual decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::~~OpenSSLServerSocketFactory ( ) [virtual]`

## 6.592.3 Member Function Documentation

6.592.3.1 `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket ( ) [virtual]`

Create a new **ServerSocket** (p. 3447) that is unbound.

The **ServerSocket** (p. 3447) will have been configured with the defaults from the factory.

### Returns

new **ServerSocket** (p. 3447) pointer that is owned by the caller.

### Exceptions

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3447) cannot be created for some reason.
--------------------	---

Reimplemented from **decaf::net::ServerSocketFactory** (p. 3458).

6.592.3.2 `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket ( int port ) [virtual]`

Create a new **ServerSocket** (p. 3447) that is bound to the given port.

The **ServerSocket** (p. 3447) will have been configured with the defaults from the factory.

### Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3447) to.
-------------	--

### Returns

new **ServerSocket** (p. 3447) pointer that is owned by the caller.

### Exceptions

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3447) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 3458).

**6.592.3.3** `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket ( int port, int backlog, const decaf::net::InetAddress * address )` [virtual]

Create a new **ServerSocket** (p. 3447) that is bound to the given port.

The **ServerSocket** (p. 3447) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3447) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 3447) will listen on all interfaces.

#### Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3447) to.
<i>backlog</i>	The number of pending connect request the <b>ServerSocket</b> (p. 3447) can queue.
<i>address</i>	The address of the interface on the local machine to bind to.

#### Returns

new **ServerSocket** (p. 3447) pointer that is owned by the caller.

#### Exceptions

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3447) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 3459).

**6.592.3.4** `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket ( int port, int backlog )` [virtual]

Create a new **ServerSocket** (p. 3447) that is bound to the given port.

The **ServerSocket** (p. 3447) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3447) will use the specified connection backlog setting.

#### Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3447) to.
<i>backlog</i>	The number of pending connect request the <b>ServerSocket</b> (p. 3447) can queue.

#### Returns

new **ServerSocket** (p. 3447) pointer that is owned by the caller.

#### Exceptions

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3447) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 3459).

**6.592.3.5** `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::getDefaultCipherSuites ( ) [virtual]`

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

#### Returns

an STL vector containing the list of cipher suites enabled by default.

#### See also

`getSupportedCipherSuites()` (p. 3670)

Implements `decaf::net::ssl::SSLServerSocketFactory` (p. 3669).

**6.592.3.6** `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::getSupportedCipherSuites ( ) [virtual]`

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

#### Returns

an STL vector containing the list of supported cipher suites.

#### See also

`getDefaultCipherSuites()` (p. 3669)

Implements `decaf::net::ssl::SSLServerSocketFactory` (p. 3670).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h`

## 6.593 `decaf::internal::net::ssl::openssl::OpenSSLSocket` Class Reference

Wraps a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocket:

## Public Member Functions

- **OpenSSLSocket** (**OpenSSLParameters** \*parameters)
- **OpenSSLSocket** (**OpenSSLParameters** \*parameters, const **decaf::net::InetAddress** \*address, int port)
- **OpenSSLSocket** (**OpenSSLParameters** \*parameters, const **decaf::net::InetAddress** \*address, int port, const **decaf::net::InetAddress** \*localAddress, int localPort)
- **OpenSSLSocket** (**OpenSSLParameters** \*parameters, const std::string &host, int port)
- **OpenSSLSocket** (**OpenSSLParameters** \*parameters, const std::string &host, int port, const **decaf::net::InetAddress** \*localAddress, int localPort)
- virtual **~OpenSSLSocket** ()
- virtual void **connect** (const std::string &host, int port, int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )

*Connects to the specified destination, with a specified timeout value.*

*If a connection to the remote host is not established within the specified timeout interval than an **SocketTimeoutException** (p. 3650) is thrown. A timeout value of zero is treated as an infinite timeout.*

### Parameters

host	<i>The host name or IP address of the remote host to connect to.</i>
port	<i>The port on the remote host to connect to.</i>
timeout	<i>The number of Milliseconds to wait before treating the connection as failed.</i>

### Exceptions

IOException	<i>Thrown if a failure occurred in the connect.</i>
<b>SocketTimeoutException</b> (p. 3650)	<i>if the timeout for connection is exceeded.</i>
IllegalArgumentEx- ception	<i>if the timeout value is negative or the endpoint is invalid.</i>

- virtual void **close** () throw ( decaf::io::IOException )

*Closes the **Socket** (p. 3607).*

*Once closed a **Socket** (p. 3607) cannot be connected or otherwise operated upon, a new **Socket** (p. 3607) instance must be created.*

### Exceptions

IOException	<i>if an I/O error occurs while closing the <b>Socket</b> (p. 3607).</i>
-------------	--

- virtual **decaf::io::InputStream** \* **getInputStream** () throw ( decaf::io::IOException )

*Gets the **InputStream** for this socket if its connected.*



The pointer returned is the property of the associated **Socket** (p. 3607) and should not be deleted by the caller.

When the returned **InputStream** is performing a blocking operation and the underlying connection is closed or otherwise broken the read calls will normally throw an exception to indicate the failure.

Closing the **InputStream** will also close the underlying **Socket** (p. 3607).

#### Returns

The **InputStream** for this socket.

#### Exceptions

IOException	if an error occurs during creation of the <b>InputStream</b> , also if the <b>Socket</b> (p. 3607) is not connected or the input has been shutdown previously.
-------------	--

- virtual **decaf::io::OutputStream \* getOutputStream ()** throw ( decaf::io::IOException )

Gets the **OutputStream** for this socket if it is connected.

The pointer returned is the property of the **Socket** (p. 3607) instance and should not be deleted by the caller.

Closing the returned **Socket** (p. 3607) will also close the underlying **Socket** (p. 3607).

#### Returns

the **OutputStream** for this socket.

#### Exceptions

IOException	if an error occurs during the creation of this <b>OutputStream</b> , or if the <b>Socket</b> (p. 3607) is closed or the output has been shutdown previously.
-------------	--

- virtual void **shutdownInput ()** throw ( decaf::io::IOException )

Shuts down the **InputStream** for this socket essentially marking it as EOF.

The stream returns EOF for any calls to read after this method has been called.

#### Exceptions

IOException	if an I/O error occurs while performing this operation.
-------------	---

- virtual void **shutdownOutput ()** throw ( decaf::io::IOException )

Shuts down the **OutputStream** for this socket, any data already written to the socket will be sent, any further calls to **OutputStream::write** will throw an **IOException**.

#### Exceptions

IOException	if an I/O error occurs while performing this operation.
-------------	---

- virtual void **setOOBInline** (bool value) throw ( decaf::net::SocketException )

Sets the value of the **OOBINLINE** for this socket, by default this option is disabled.

If enabled the urgent data is read inline on the **Socket**'s **InputStream**, no notification is give.

#### Returns

true if **OOBINLINE** is enabled, false otherwise.

*Exceptions*

<b>SocketException</b> (p. 3627)	<i>if an error is encountered while performing this operation.</i>
-------------------------------------	--

- virtual void **sendUrgentData** (int data) throw ( decaf::io::IOException )

*Sends on byte of urgent data to the **Socket** (p. 3607).*

*Parameters*

data	<i>The value to write as urgent data, only the lower eight bits are sent.</i>
------	---

*Exceptions*

IOException	<i>if an I/O error occurs while performing this operation.</i>
-------------	--

- virtual std::vector< std::string > **getSupportedCipherSuites** () const

*Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 3670).*

*Normally not all of these cipher suites will be enabled on the **Socket** (p. 3607).*

*Returns*

*a vector containing the names of all the supported cipher suites.*

- virtual std::vector< std::string > **getSupportedProtocols** () const

*Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 3670) instance.*

*Returns*

*a vector containing the names of all the supported protocols.*

- virtual std::vector< std::string > **getEnabledCipherSuites** () const

*Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSL Socket** (p. 3607).*

*Returns*

*vector of the names of all enabled Cipher Suites.*

- virtual void **setEnabledCipherSuites** (const std::vector< std::string > &suites)

*Sets the Cipher Suites that are to be enabled on the **SSL Socket** (p. 3607) connection. Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.*

*Parameters*

suites	<i>An Vector of names for all the Cipher Suites that are to be enabled.</i>
--------	---

*Exceptions*

IllegalArgumentExcep- tion	<i>if the vector is empty or one of the names is invalid.</i>
-------------------------------	---

- virtual std::vector< std::string > **getEnabledProtocols** () const

Returns a vector containing the names of all the currently enabled Protocols for this **SSL Socket** (p. 3607).

**Returns**

vector of the names of all enabled Protocols.

- virtual void **setEnabledProtocols** (const std::vector< std::string > &protocols)

Sets the Protocols that are to be enabled on the **SSL Socket** (p. 3607) connection. Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

**Parameters**

protocols	An Vector of names for all the Protocols that are to be enabled.
-----------	--

**Exceptions**

IllegalArgumentExcep- tion	if the vector is empty or one of the names is invalid.
-------------------------------	--

- virtual void **startHandshake** ()

Initiates a handshake for this **SSL Connection**, this can be necessary for several reasons such as using new encryption keys, or starting a new session.

When called for the first time after the socket connects this method blocks until the handshake is completed. The provider is not require to support multiple handshakes and can throw an **IOException** to indicate an error.

**Exceptions**

IOException	if an I/O error occurs while performing the Handshake
-------------	---

- virtual void **setUseClientMode** (bool value)

Determines the mode that the socket uses when a handshake is initiated, client or server.

This method must be called prior to any handshake attempts on this **Socket** (p. 3607), once a handshake has be initiated this socket remains the the set mode; client or server; for the life of this object.

**Parameters**

value	The mode setting, true for client or false for server.
-------	--

**Exceptions**

IllegalArguementEx- ception	if the handshake process has begun and mode is locked.
--------------------------------	--

- virtual bool **getUseClientMode** () const

Gets whether this **Socket** (p. 3607) is in Client or Server mode, true indicates that the mode is set to Client.

**Returns**

true if the **Socket** (p. 3607) is in Client mode, false otherwise.

- virtual void **setNeedClientAuth** (bool value)

*Sets the **Socket** (p. 3607) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*

*This option only applies to sockets in the Server mode.*

*If the option is enabled and the client does not provide a certificate then the handshake is considered failed and the connection is refused. Calling this method resets any previous value for this option as well as clears any value set in the setWantClientAuth method.*

**Parameters**

value	The value indicating if a client is required to authenticate itself or not.
-------	---

- virtual bool **getNeedClientAuth** () const

*Returns if this socket is configured to require client authentication, true means that it has and that clients that failed to authenticate will be rejected.*

*This option is only useful when the socket is operating in server mode.*

**Returns**

*true if client authentication is required.*

- virtual void **setWantClientAuth** (bool value)

*Sets the **Socket** (p. 3607) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*

*This option only applies to sockets in the Server mode.*

*If the option is enabled and the client does not provide a certificate then the handshake is considered to have succeeded, if it does send a certificate and that certificate is invalid the the handshake will fail. Calling this method resets any previous value for this option as well as clears any value set in the setNeedClientAuth method.*

**Parameters**

value	The value indicating if a client is requested to authenticate itself or not.
-------	--

- virtual bool **getWantClientAuth** () const

*Returns if this socket is configured to request client authentication, true means that it has and that clients that failed to authenticate will be rejected but that clients that do not send a certificate are not considered to have failed authentication.*

*This option is only useful when the socket is operating in server mode.*

**Returns**

*true if client authentication is required.*

- int **read** (unsigned char \*buffer, int size, int offset, int length)

*Reads the requested data from the Socket and write it into the passed in buffer.*

- void **write** (const unsigned char \*buffer, int size, int offset, int length)

*Writes the specified data in the passed in buffer to the Socket.*

- int **available** ()

*Gets the number of bytes in the Socket buffer that can be read without blocking.*

### 6.593.1 Detailed Description

Wraps a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.

**Since**

1.0

### 6.593.2 Constructor & Destructor Documentation

**6.593.2.1** decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (   
 OpenSSLParameters \* *parameters* )

**6.593.2.2** decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (   
 OpenSSLParameters \* *parameters*, const decaf::net::InetAddress \* *address*, int *port* )

**6.593.2.3** decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (   
 OpenSSLParameters \* *parameters*, const decaf::net::InetAddress \* *address*, int *port*, const decaf::net::InetAddress \* *localAddress*, int *localPort* )

**6.593.2.4** decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (   
 OpenSSLParameters \* *parameters*, const std::string & *host*, int *port* )

**6.593.2.5** decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (   
 OpenSSLParameters \* *parameters*, const std::string & *host*, int *port*, const decaf::net::InetAddress \* *localAddress*, int *localPort* )

**6.593.2.6** virtual decaf::internal::net::ssl::openssl::OpenSSLSocket::~~OpenSSLSocket ( )   
 [virtual]

### 6.593.3 Member Function Documentation

**6.593.3.1** int decaf::internal::net::ssl::openssl::OpenSSLSocket::available ( )

Gets the number of bytes in the Socket buffer that can be read without blocking.

**Returns**

the number of bytes that can be read from the Socket without blocking.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

**6.593.3.2** `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::close ( ) throw ( decaf::io::IOException )` [virtual]

Closes the **Socket** (p. 3607).

Once closed a **Socket** (p. 3607) cannot be connected or otherwise operated upon, a new **Socket** (p. 3607) instance must be created.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while closing the <b>Socket</b> (p. 3607).
--------------------	---

Reimplemented from **decaf::net::Socket** (p. 3614).

**6.593.3.3** `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::connect ( const std::string & host, int port, int timeout ) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )` [virtual]

Connects to the specified destination, with a specified timeout value.

If a connection to the remote host is not established within the specified timeout interval than an **SocketTimeoutException** (p. 3650) is thrown. A timeout value of zero is treated as an infinite timeout.

#### Parameters

<i>host</i>	The host name or IP address of the remote host to connect to.
<i>port</i>	The port on the remote host to connect to.
<i>timeout</i>	The number of Milliseconds to wait before treating the connection as failed.

#### Exceptions

<i>IOException</i>	Thrown if a failure occurred in the connect.
<i>SocketTimeoutException</i> (p. 3650)	if the timeout for connection is exceeded.
<i>IllegalArgumentEx-ception</i>	if the timeout value is negative or the endpoint is invalid.

Reimplemented from **decaf::net::Socket** (p. 3615).

**6.593.3.4** `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getEnabledCipherSuites ( ) const` [virtual]

Returns a vector containing the names of all the currently enabled Cipher Suites for this SSL **Socket** (p. 3607).

#### Returns

vector of the names of all enabled Cipher Suites.

Implements **decaf::net::ssl::SSLSocket** (p. 3674).

**6.593.3.5** `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getEnabledProtocols ( ) const [virtual]`

Returns a vector containing the names of all the currently enabled Protocols for this **SSL Socket** (p. 3607).

#### Returns

vector of the names of all enabled Protocols.

Implements **decaf::net::ssl::SSLSocket** (p. 3674).

**6.593.3.6** `virtual decaf::io::InputStream* decaf::internal::net::ssl::openssl::OpenSSLSocket::getInputStream ( ) throw ( decaf::io::IOException ) [virtual]`

Gets the InputStream for this socket if its connected.

The pointer returned is the property of the associated **Socket** (p. 3607) and should not be deleted by the caller.

When the returned InputStream is performing a blocking operation and the underlying connection is closed or otherwise broken the read calls will normally throw an exception to indicate the failure.

Closing the InputStream will also close the underlying **Socket** (p. 3607).

#### Returns

The InputStream for this socket.

#### Exceptions

<i>IOException</i>	if an error occurs during creation of the InputStream, also if the <b>Socket</b> (p. 3607) is not connected or the input has been shutdown previously.
--------------------	--

Reimplemented from **decaf::net::Socket** (p. 3616).

**6.593.3.7** `virtual bool decaf::internal::net::ssl::openssl::OpenSSLSocket::getNeedClientAuth ( ) const [virtual]`

Returns if this socket is configured to require client authentication, true means that it has and that clients that failed to authenticate will be rejected.

This option is only useful when the socket is operating in server mode.

**Returns**

true if client authentication is required.

Implements **decaf::net::ssl::SSLSocket** (p. 3675).

**6.593.3.8** `virtual decaf::io::OutputStream* decaf::internal::net::ssl::openssl::OpenSSLSocket::getOutputStream ( ) throw ( decaf::io::IOException ) [virtual]`

Gets the OutputStream for this socket if it is connected.

The pointer returned is the property of the **Socket** (p. 3607) instance and should not be deleted by the caller.

Closing the returned **Socket** (p. 3607) will also close the underlying **Socket** (p. 3607).

**Returns**

the OutputStream for this socket.

**Exceptions**

<i>IOException</i>	if an error occurs during the creation of this OutputStream, or if the <b>Socket</b> (p. 3607) is closed or the output has been shutdown previously.
--------------------	--

Reimplemented from **decaf::net::Socket** (p. 3617).

**6.593.3.9** `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getSupportedCipherSuites ( ) const [virtual]`

Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 3670).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 3607).

**Returns**

a vector containing the names of all the supported cipher suites.

Implements **decaf::net::ssl::SSLSocket** (p. 3675).

**6.593.3.10** `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getSupportedProtocols ( ) const [virtual]`

Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 3670) instance.



**Returns**

a vector containing the names of all the supported protocols.

Implements **decaf::net::ssl::SSLSocket** (p. 3675).

**6.593.3.11** `virtual bool decaf::internal::net::ssl::openssl::OpenSSLSocket::getUseClientMode ( ) const` [virtual]

Gets whether this **Socket** (p. 3607) is in Client or Server mode, true indicates that the mode is set to Client.

**Returns**

true if the **Socket** (p. 3607) is in Client mode, false otherwise.

Implements **decaf::net::ssl::SSLSocket** (p. 3676).

**6.593.3.12** `virtual bool decaf::internal::net::ssl::openssl::OpenSSLSocket::getWantClientAuth ( ) const` [virtual]

Returns if this socket is configured to request client authentication, true means that it has and that clients that failed to authenticate will be rejected but that clients that do not send a certificate are not considered to have failed authentication.

This option is only useful when the socket is operating in server mode.

**Returns**

true if client authentication is required.

Implements **decaf::net::ssl::SSLSocket** (p. 3676).

**6.593.3.13** `int decaf::internal::net::ssl::openssl::OpenSSLSocket::read ( unsigned char * buffer, int size, int offset, int length )`

Reads the requested data from the Socket and write it into the passed in buffer.

**Parameters**

<i>buffer</i>	The buffer to read into
<i>size</i>	The size of the specified buffer
<i>offset</i>	The offset into the buffer where reading should start filling.
<i>length</i>	The number of bytes past offset to fill with data.

**Returns**

the actual number of bytes read or -1 if at EOF.

**Exceptions**

<i>IOException</i>	if an I/O error occurs during the read.
<i>NullPointerException</i>	if buffer is Null.
<i>IndexOutOfBoundsException</i>	if offset + length is greater than buffer size.

**6.593.3.14** `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::sendUrgentData ( int data ) throw ( decaf::io::IOException ) [virtual]`

Sends on byte of urgent data to the **Socket** (p. 3607).

**Parameters**

<i>data</i>	The value to write as urgent data, only the lower eight bits are sent.
-------------	--

**Exceptions**

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Reimplemented from **decaf::net::Socket** (p. 3621).

**6.593.3.15** `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setEnabledCipherSuites ( const std::vector< std::string > & suites ) [virtual]`

Sets the Cipher Suites that are to be enabled on the SSL **Socket** (p. 3607) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

**Parameters**

<i>suites</i>	An Vector of names for all the Cipher Suites that are to be enabled.
---------------	--

**Exceptions**

<i>IllegalArgumentEx-ception</i>	if the vector is empty or one of the names is invalid.
----------------------------------	--

Implements **decaf::net::ssl::SSLSocket** (p. 3676).

**6.593.3.16** `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setEnabledProtocols ( const std::vector< std::string > & protocols ) [virtual]`

Sets the Protocols that are to be enabled on the SSL **Socket** (p. 3607) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

## 6.593 decaf::internal::net::ssl::openssl::OpenSSLSocket Class Reference 2955

### Parameters

<i>protocols</i>	An Vector of names for all the Protocols that are to be enabled.
------------------	--

### Exceptions

<i>IllegalArgumentException</i>	if the vector is empty or one of the names is invalid.
---------------------------------	--

Implements **decaf::net::ssl::SSLSocket** (p. 3677).

**6.593.3.17** virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setNeedClientAuth ( bool *value* ) [virtual]

Sets the **Socket** (p. 3607) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled an the client does not provide a certificate then the handshake is considered failed and the connection is refused. Calling this method resets any previous value for this option as well as clears any value set in the setWantClientAuth method.

### Parameters

<i>value</i>	The value indicating if a client is required to authenticate itself or not.
--------------	---

Implements **decaf::net::ssl::SSLSocket** (p. 3677).

**6.593.3.18** virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setOOBInline ( bool *value* ) throw ( decaf::net::SocketException ) [virtual]

Sets the value of the OOBINLINE for this socket, by default this option is disabled.

If enabled the urgent data is read inline on the Socket's InputStream, no notification is give.

### Returns

true if OOBINLINE is enabled, false otherwise.

### Exceptions

<i>SocketException</i> (p. 3627)	if an error is encountered while performing this operation.
-------------------------------------	---

Reimplemented from **decaf::net::Socket** (p. 3621).

**6.593.3.19** `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setUseClientMode ( bool value ) [virtual]`

Determines the mode that the socket uses when a handshake is initiated, client or server.

This method must be called prior to any handshake attempts on this **Socket** (p. 3607), once a handshake has been initiated this socket remains in the set mode; client or server, for the life of this object.

#### Parameters

<i>value</i>	The mode setting, true for client or false for server.
--------------	--

#### Exceptions

<i>IllegalArgumentException</i>	if the handshake process has begun and mode is locked.
---------------------------------	--

Implements **decaf::net::ssl::SSLSocket** (p. 3678).

**6.593.3.20** `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setWantClientAuth ( bool value ) [virtual]`

Sets the **Socket** (p. 3607) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled and the client does not provide a certificate then the handshake is considered to have succeeded, if it does send a certificate and that certificate is invalid the handshake will fail. Calling this method resets any previous value for this option as well as clears any value set in the `setNeedClientAuth` method.

#### Parameters

<i>value</i>	The value indicating if a client is requested to authenticate itself or not.
--------------	--

Implements **decaf::net::ssl::SSLSocket** (p. 3678).

**6.593.3.21** `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::shutdownInput ( ) throw ( decaf::io::IOException ) [virtual]`

Shuts down the `InputStream` for this socket essentially marking it as EOF.

The stream returns EOF for any calls to read after this method has been called.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Reimplemented from **decaf::net::Socket** (p. 3625).

**6.593.3.22** virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::shutdownOutput ( )  
throw ( decaf::io::IOException ) [virtual]

Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to OuputStream::write will throw an IOException.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Reimplemented from **decaf::net::Socket** (p. 3625).

**6.593.3.23** virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::startHandshake ( )  
[virtual]

Initiates a handshake for this SSL Connection, this can be necessary for several reasons such as using new encryption keys, or starting a new session.

When called for the first time after the socket connects this method blocks until the handshake is completed. The provider is not require to support multiple handshakes and can throw an IOException to indicate an error.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while performing the Handshake
--------------------	---

Implements **decaf::net::ssl::SSLSocket** (p. 3679).

**6.593.3.24** void decaf::internal::net::ssl::openssl::OpenSSLSocket::write ( const unsigned char  
\* *buffer*, int *size*, int *offset*, int *length* )

Writes the specified data in the passed in buffer to the Socket.

#### Parameters

<i>buffer</i>	The buffer to write to the socket.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset into the buffer where the data to write starts at.
<i>length</i>	The number of bytes past offset to write.

#### Exceptions

<i>IOException</i>	if an I/O error occurs during the write.
<i>NullPointerException</i>	if buffer is Null.
<i>IndexOutOfBoundsException</i>	if offset + length is greater than buffer size.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h`

## 6.594 decaf::internal::net::ssl::openssl::OpenSSLSocketException

### Class Reference

Subclass of the standard `SocketException` that knows how to produce an error message from the OpenSSL error stack.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h>
```

Inheritance diagram for `decaf::internal::net::ssl::openssl::OpenSSLSocketException`:

### Public Member Functions

- **OpenSSLSocketException** () throw ()  
*Creates a new **OpenSSLSocketException** (p. 2955) with default values.*
- **OpenSSLSocketException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **OpenSSLSocketException** (const **OpenSSLSocketException** &ex) throw ()  
*Copy Constructor.*
- **OpenSSLSocketException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Create a new **OpenSSLSocketException** (p. 2955) and initializes the file name and line number where this message occurred.*
- **OpenSSLSocketException** (const std::exception \*cause) throw ()  
*Creates a new **OpenSSLSocketException** (p. 2955) with the passed exception set as the cause of this exception.*
- **OpenSSLSocketException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Create a new **OpenSSLSocketException** (p. 2955) and initializes the file name and line number where this message occurred.*
- **OpenSSLSocketException** (const char \*file, const int lineNumber) throw ()  
*Create a new **OpenSSLSocketException** (p. 2955) and initializes the file name and line number where this message occurred.*
- virtual **OpenSSLSocketException** \* clone () const  
*Clones this exception.*
- virtual ~**OpenSSLSocketException** () throw ()

## Protected Member Functions

- std::string **getErrorString** () const

*Gets and formats an error message string from the OpenSSL error stack.*

### 6.594.1 Detailed Description

Subclass of the standard SocketException that knows how to produce an error message from the OpenSSL error stack.

Since

1.0

### 6.594.2 Constructor & Destructor Documentation

6.594.2.1 decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException  
( ) throw ()

Creates an new **OpenSSLSocketException** (p. 2955) with default values.

6.594.2.2 decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException  
( const Exception & ex ) throw ()

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An Exception object that should become this type of Exception.
-----------	--

6.594.2.3 decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException  
( const OpenSSLSocketException & ex ) throw ()

Copy Constructor.

#### Parameters

<i>ex</i>	The <b>OpenSSLSocketException</b> (p. 2955) whose values should be copied to this instance.
-----------	---

**6.594.2.4** `decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()`

Create a new **OpenSSLSocketException** (p. 2955) and initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message.

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown (can be null).
<i>msg</i>	The error message to report.
...	The list of primitives that are formatted into the message.

**6.594.2.5** `decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException ( const std::exception * cause ) throw ()`

Creates a new **OpenSSLSocketException** (p. 2955) with the passed exception set as the cause of this exception.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.594.2.6** `decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()`

Create a new **OpenSSLSocketException** (p. 2955) and initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message.

#### Parameters

<i>file</i>	The file name where exception occurs.
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The error message to report.
...	The list of primitives that are formatted into the message



**6.594.2.7** decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException  
( const char \* *file*, const int *lineNumber* ) throw ()

Create a new **OpenSSLSocketException** (p. 2955) and initializes the file name and line number where this message occurred.

Sets the message to report by getting the complete set of error messages from the OpenSSL error stack and concatenating them into one string.

#### Parameters

<i>file</i>	The file name where exception occurs.
<i>lineNumber</i>	The line number where the exception occurred.

**6.594.2.8** virtual decaf::internal::net::ssl::openssl::OpenSSLSocketException::~OpenSSLSocketException  
( ) throw () [virtual]

#### 6.594.3 Member Function Documentation

**6.594.3.1** virtual OpenSSLSocketException\* decaf::internal::net::ssl::openssl::OpenSSLSocketException::clone (  
) const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override this method.

#### Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 3629).

**6.594.3.2** std::string decaf::internal::net::ssl::openssl::OpenSSLSocketException::getErrorString  
( ) const [protected]

Gets and formats an error message string from the OpenSSL error stack.

#### Returns

a string containing the complete OpenSSL error string.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLSocketException.h**

## 6.595 decaf::internal::net::ssl::openssl::OpenSSLSocketFactory Class Reference

Client Socket Factory that creates SSL based client sockets using the OpenSSL library.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocketFactory:

### Public Member Functions

- **OpenSSLSocketFactory** (**OpenSSLContextSpi** \*parent)
- virtual **~OpenSSLSocketFactory** ()
- virtual **decaf::net::Socket** \* **createSocket** () throw ( decaf::io::IOException )

*Creates an unconnected **Socket** (p. 3607) object.*

#### Returns

*a new **Socket** (p. 3607) object, caller must free this object when done.*

#### Exceptions

IOException	<i>if the <b>Socket</b> (p. 3607) cannot be created.</i>
-------------	--

- virtual **decaf::net::Socket** \* **createSocket** (const **decaf::net::InetAddress** \*host, int port) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

*Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).*

#### Parameters

host	<i>The host to connect the socket to.</i>
port	<i>The port on the remote host to connect to.</i>

#### Returns

*a new **Socket** (p. 3607) object, caller must free this object when done.*

#### Exceptions

IOException	<i>if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.</i>
<b>UnknownHostException</b> (p. 4019)	<i>if the host name is not known.</i>

- virtual **decaf::net::Socket** \* **createSocket** (const **decaf::net::InetAddress** \*host, int port, const **decaf::net::InetAddress** \*ifAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

*Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).*

*The **Socket** (p. 3607) will be bound to the specified local address and port.*

## 6.595 decaf::internal::net::ssl::openssl::OpenSSLSocketFactory Class Reference

### Parameters

host	The host to connect the socket to.
port	The port on the remote host to connect to.
ifAddress	The address on the local machine to bind the <b>Socket</b> (p. 3607) to.
localPort	The local port to bind the <b>Socket</b> (p. 3607) to.

### Returns

a new **Socket** (p. 3607) object, caller must free this object when done.

### Exceptions

IOException	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<b>UnknownHostException</b> (p. 4019)	if the host name is not known.

- virtual **decaf::net::Socket** \* **createSocket** (const std::string &hostname, int port) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).

### Parameters

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.

### Returns

a new **Socket** (p. 3607) object, caller must free this object when done.

### Exceptions

IOException	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<b>UnknownHostException</b> (p. 4019)	if the host name is not known.

- virtual **decaf::net::Socket** \* **createSocket** (const std::string &name, int port, const **decaf::net::InetAddress** \*ifAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).

### Parameters

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.
ifAddress	The address on the local machine to bind the <b>Socket</b> (p. 3607) to.
localPort	The local port to bind the <b>Socket</b> (p. 3607) to.

### Returns

a new **Socket** (p. 3607) object, caller must free this object when done.

### Exceptions

IOException	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
-------------	---

<b>UnknownHostException</b> (p. 4019)	if the host name is not known.
--	--------------------------------

- virtual `std::vector< std::string > getDefaultCipherSuites ()`

Returns the list of cipher suites which are enabled by default.  
Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

**Returns**

an STL vector containing the list of cipher suites enabled by default.

**See also**

`getSupportedCipherSuites()` (p. 3682)

- virtual `std::vector< std::string > getSupportedCipherSuites ()`

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

**Returns**

an STL vector containing the list of supported cipher suites.

**See also**

`getDefaultCipherSuites()` (p. 3681)

- virtual `decaf::net::Socket * createSocket (decaf::net::Socket *socket, std::string host, int port, bool autoClose)`

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

**Parameters**

socket	The existing socket to layer over.
host	The server host the original <b>Socket</b> (p. 3607) is connected to.
port	The server port the original <b>Socket</b> (p. 3607) is connected to.
autoClose	Should the layered over <b>Socket</b> (p. 3607) be closed when the topmost socket is closed.

**Returns**

a new **Socket** (p. 3607) instance that wraps the given **Socket** (p. 3607).

**Exceptions**

IOException	if an I/O exception occurs while performing this operation.
-------------	---

## 6.595 decaf::internal::net::ssl::openssl::OpenSSLSocketFactory Class Reference

<b>UnknownHostException</b> (p. 4019)	<i>if the host is unknown.</i>
--	--------------------------------

### 6.595.1 Detailed Description

Client Socket Factory that creates SSL based client sockets using the OpenSSL library.

#### Since

1.0

### 6.595.2 Constructor & Destructor Documentation

6.595.2.1 **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::OpenSSLSocketFactory ( OpenSSLContextSpi \* *parent* )**

6.595.2.2 **virtual decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::~~OpenSSLSocketFactory ( ) [virtual]**

### 6.595.3 Member Function Documentation

6.595.3.1 **virtual decaf::net::Socket\* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket ( ) throw ( decaf::io::IOException ) [virtual]**

Creates an unconnected **Socket** (p. 3607) object.

#### Returns

a new **Socket** (p. 3607) object, caller must free this object when done.

#### Exceptions

<i>IOException</i>	if the <b>Socket</b> (p. 3607) cannot be created.
--------------------	---

Reimplemented from **decaf::net::SocketFactory** (p. 3631).

6.595.3.2 **virtual decaf::net::Socket\* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket ( decaf::net::Socket \* *socket*, std::string *host*, int *port*, bool *autoClose* ) [virtual]**

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer

destination. This socket is configured using the socket options established for this factory.

#### Parameters

<i>socket</i>	The existing socket to layer over.
<i>host</i>	The server host the original <b>Socket</b> (p. 3607) is connected to.
<i>port</i>	The server port the original <b>Socket</b> (p. 3607) is connected to.
<i>autoClose</i>	Should the layered over <b>Socket</b> (p. 3607) be closed when the topmost socket is closed.

#### Returns

a new **Socket** (p. 3607) instance that wraps the given **Socket** (p. 3607).

#### Exceptions

<i>IOException</i>	if an I/O exception occurs while performing this operation.
<b>UnknownHostException</b> (p. 4019)	if the host is unknown.

Implements **decaf::net::ssl::SSLSocketFactory** (p. 3680).

**6.595.3.3** `virtual decaf::net::Socket* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket  
( const decaf::net::InetAddress * host, int port ) throw ( de-  
caf::io::IOException, decaf::net::UnknownHostException )  
[virtual]`

Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).

#### Parameters

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

#### Returns

a new **Socket** (p. 3607) object, caller must free this object when done.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<b>UnknownHostException</b> (p. 4019)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3631).

## 6.595 decaf::internal::net::ssl::openssl::OpenSSLSocketFactory Class Reference

**6.595.3.4** virtual decaf::net::Socket\* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket ( const std::string & *hostname*, int *port* ) throw ( decaf::io::IOException, decaf::net::UnknownHostException ) [virtual]

Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).

### Parameters

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

### Returns

a new **Socket** (p. 3607) object, caller must free this object when done.

### Exceptions

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<b>UnknownHostException</b> (p. 4019)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3632).

**6.595.3.5** virtual decaf::net::Socket\* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket ( const std::string & *name*, int *port*, const decaf::net::InetAddress \* *ifAddress*, int *localPort* ) throw ( decaf::io::IOException, decaf::net::UnknownHostException ) [virtual]

Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).

### Parameters

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the <b>Socket</b> (p. 3607) to.
<i>localPort</i>	The local port to bind the <b>Socket</b> (p. 3607) to.

### Returns

a new **Socket** (p. 3607) object, caller must free this object when done.

### Exceptions

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<b>UnknownHostException</b> (p. 4019)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3632).

```
6.595.3.6 virtual decaf::net::Socket* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket
( const decaf::net::InetAddress * host, int port, const de-
caf::net::InetAddress * ifAddress, int localPort ) throw (
decaf::io::IOException, decaf::net::UnknownHostException )
[virtual]
```

Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).

The **Socket** (p. 3607) will be bound to the specified local address and port.

#### Parameters

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the <b>Socket</b> (p. 3607) to.
<i>localPort</i>	The local port to bind the <b>Socket</b> (p. 3607) to.

#### Returns

a new **Socket** (p. 3607) object, caller must free this object when done.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<i>UnknownHostException</i> (p. 4019)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3633).

```
6.595.3.7 virtual std::vector<std::string> de-
caf::internal::net::ssl::openssl::OpenSSLSocketFactory::getDefaultCipherSuites ( )
[virtual]
```

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

#### Returns

an STL vector containing the list of cipher suites enabled by default.

#### See also

**getSupportedCipherSuites()** (p. 3682)

Implements **decaf::net::ssl::SSLSocketFactory** (p. 3681).



6.595.3.8 virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::getSupportedCipherSuites ( ) [virtual]

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

### Returns

an STL vector containing the list of supported cipher suites.

### See also

**getDefaultCipherSuites()** (p. 3681)

Implements **decaf::net::ssl::SSLSocketFactory** (p. 3682).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h

## 6.596 decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream Class Reference

An output stream for reading data from an OpenSSL Socket instance.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream:

### Public Member Functions

- **OpenSSLSocketInputStream (OpenSSLSocket \*socket)**
- virtual **~OpenSSLSocketInputStream ()**
- virtual int **available ()** const throw ( decaf::io::IOException )

*Indicates the number of bytes available.*

*The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.*

*The default implementation of this method returns zero.*

#### Returns

*the number of bytes available on this input stream.*

*Exceptions*

<b>IOException</b> (p. 2210)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

- virtual void **close** () throw ( decaf::io::IOException )

*Close - does nothing.*

- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::Unsupported )

*Not supported.*

**Protected Member Functions**

- virtual int **doReadByte** () throw ( io::IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

**6.596.1 Detailed Description**

An output stream for reading data from an OpenSSL Socket instance.

Since

1.0

**6.596.2 Constructor & Destructor Documentation**

**6.596.2.1** decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::OpenSSLSocketInputStream ( OpenSSLSocket \* *socket* )

**6.596.2.2** virtual decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::~~OpenSSLSocketInputStream ( ) [virtual]

**6.596.3 Member Function Documentation**

**6.596.3.1** virtual int decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::available ( ) const throw ( decaf::io::IOException ) [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

## Returns

the number of bytes available on this input stream.

## Exceptions

<i><b>IOException</b></i> (p. 2210)	if an I/O error occurs.
--	-------------------------

Reimplemented from **decaf::io::InputStream** (p. 2107).

**6.596.3.2** `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::close ( )  
throw ( decaf::io::IOException ) [virtual]`

Close - does nothing.

It is the responsibility of the owner of the socket object to close it.

Closes the **InputStream** (p. 2105) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::InputStream** (p. 2108).

**6.596.3.3** `virtual int decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::doReadArrayBounded  
( unsigned char * buffer, int size, int offset, int  
length ) throw ( decaf::io::IOException, de-  
caf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::NullPointerException ) [protected,  
virtual]`

Reimplemented from **decaf::io::InputStream** (p. 2108).

**6.596.3.4** `virtual int decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::doReadByte  
( ) throw ( io::IOException ) [protected, virtual]`

Implements **decaf::io::InputStream** (p. 2109).

**6.596.3.5** `virtual long long decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::skip  
( long long num ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Not supported.

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions;

reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2105) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

#### Returns

total bytes skipped

#### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
<b><i>UnsupportedOperationException</i></b>	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 2113).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h

## 6.597 decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream Class Reference

OutputStream implementation used to write data to an **OpenSSLSocket** (p. 2940) instance.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream:

#### Public Member Functions

- **OpenSSLSocketOutputStream** (**OpenSSLSocket** \*socket)
- virtual **~OpenSSLSocketOutputStream** ()
- virtual void **close** () throw ( decaf::io::IOException )

*Closes this object and deallocates the appropriate resources.  
The object is generally no longer usable after calling close.*

#### Exceptions

<b>IOException</b> (p. 2210)	if an error occurs while closing.
------------------------------	-----------------------------------

*The default implementation of this method does nothing.*

## Protected Member Functions

- virtual void **doWriteByte** (unsigned char c) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### 6.597.1 Detailed Description

OutputStream implementation used to write data to an **OpenSSLSocket** (p. 2940) instance.

Since

1.0

### 6.597.2 Constructor & Destructor Documentation

6.597.2.1 decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::OpenSSLSocketOutputStream ( **OpenSSLSocket** \* *socket* )

6.597.2.2 virtual decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::~~OpenSSLSocketOutputStream ( ) [virtual]

### 6.597.3 Member Function Documentation

6.597.3.1 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::close ( ) throw ( **decaf::io::IOException** ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

## Exceptions

<b>IOException</b> (p. 2210)	if an error occurs while closing.
---------------------------------	-----------------------------------

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2993).

6.597.3.2 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::doWriteArrayBounded ( const unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [protected, virtual]

Reimplemented from `decaf::io::OutputStream` (p. 2994).

6.597.3.3 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::doWriteByte ( unsigned char c ) throw ( decaf::io::IOException )` [protected, virtual]

Implements `decaf::io::OutputStream` (p. 2994).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h`

## 6.598 activemq::wireformat::openwire::OpenWireFormat Class Reference

```
#include <src/main/activemq/wireformat/openwire/OpenWireFormat.h>
```

Inheritance diagram for `activemq::wireformat::openwire::OpenWireFormat`:

### Public Member Functions

- **OpenWireFormat** (const `decaf::util::Properties` &properties)  
*Constructs a new **OpenWireFormat** (p. 2971) object.*
- virtual `~OpenWireFormat` ()
- virtual bool **hasNegotiator** () const  
*Returns true if this **WireFormat** (p. 4088) has a Negotiator that needs to wrap the Transport that uses it.*
- virtual `Pointer< transport::Transport > createNegotiator` (const `Pointer< transport::Transport > &transport`) throw ( `decaf::lang::exceptions::UnsupportedOperationException` )  
*If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.*
- void **addMarshaller** (`marshal::DataStreamMarshaller` \*marshaller)  
*Allows an external source to add marshalers to this object for types that may be marshaled or unmarshaled.*

- virtual void **marshal** (const **Pointer**< **commands::Command** > &command, const **activemq::transport::Transport** \*transport, **decaf::io::DataOutputStream** \*out) throw ( **decaf::io::IOException** )  
*Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.*
- virtual **Pointer**< **commands::Command** > **unmarshal** (const **activemq::transport::Transport** \*transport, **decaf::io::DataInputStream** \*in) throw ( **decaf::io::IOException** )  
*Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and un-marshaled into the correct form.*
- virtual int **tightMarshalNestedObject1** (**commands::DataStructure** \*object, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Utility method for Tight Marshaling the given object to the boolean stream passed.*
- void **tightMarshalNestedObject2** (**commands::DataStructure** \*o, **decaf::io::DataOutputStream** \*ds, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Utility method that will Tight marshal some internally nested object that implements the DataStructure interface.*
- **commands::DataStructure** \* **tightUnmarshalNestedObject** (**decaf::io::DataInputStream** \*dis, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Utility method used to Unmarshal a Nested DataStructure type object from the given DataInputStream.*
- **commands::DataStructure** \* **looseUnmarshalNestedObject** (**decaf::io::DataInputStream** \*dis) throw ( **decaf::io::IOException** )  
*Utility method to unmarshal an DataStructure object from an DataInputStream using the Loose Unmarshaling format.*
- void **looseMarshalNestedObject** (**commands::DataStructure** \*o, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Utility method to loosely Marshal an object that is derived from the DataStrucutre interface.*
- void **renegotiateWireFormat** (const **commands::WireFormatInfo** &info) throw ( **decaf::lang::exceptions::IllegalStateException** )  
*Called to re-negotiate the settings for the WireFormatInfo, these determine how the client and broker communicate.*
- virtual void **setPreferredWireFormatInfo** (const **Pointer**< **commands::WireFormatInfo** > &info) throw ( **decaf::lang::exceptions::IllegalStateException** )  
*Configures this object using the provided WireformatInfo object.*
- virtual const **Pointer**< **commands::WireFormatInfo** > & **getPreferredWireFormatInfo** () const  
*Gets the Preferred WireFormatInfo object that this class holds.*

- **bool isStackTraceEnabled () const**  
*Checks if the stackTraceEnabled flag is on.*
- **void setStackTraceEnabled (bool stackTraceEnabled)**  
*Sets if the stackTraceEnabled flag is on.*
- **bool isTcpNoDelayEnabled () const**  
*Checks if the tcpNoDelayEnabled flag is on.*
- **void setTcpNoDelayEnabled (bool tcpNoDelayEnabled)**  
*Sets if the tcpNoDelayEnabled flag is on.*
- **int getVersion () const**  
*Get the current Wireformat Version.*
- **void setVersion (int version) throw ( decaf::lang::exceptions::IllegalArgumentException )**  
*Set the current Wireformat Version.*
- **virtual bool inReceive () const**  
*Is there a Message being unmarshaled?*
- **bool isCacheEnabled () const**  
*Checks if the cacheEnabled flag is on.*
- **void setCacheEnabled (bool cacheEnabled)**  
*Sets if the cacheEnabled flag is on.*
- **int getCacheSize () const**  
*Returns the currently set Cache size.*
- **void setCacheSize (int value)**  
*Sets the current Cache size.*
- **bool isTightEncodingEnabled () const**  
*Checks if the tightEncodingEnabled flag is on.*
- **void setTightEncodingEnabled (bool tightEncodingEnabled)**  
*Sets if the tightEncodingEnabled flag is on.*
- **bool isSizePrefixDisabled () const**  
*Checks if the sizePrefixDisabled flag is on.*
- **void setSizePrefixDisabled (bool sizePrefixDisabled)**  
*Sets if the sizePrefixDisabled flag is on.*



## 6.598 activemq::wireformat::openwire::OpenWireFormat Class Reference 2977

- long long **getMaxInactivityDuration** () const  
*Gets the MaxInactivityDuration setting.*
- void **setMaxInactivityDuration** (long long value)  
*Sets the MaxInactivityDuration setting.*
- long long **getMaxInactivityDurationInitialDelay** () const  
*Gets the MaxInactivityDurationInitialDelay setting.*
- void **setMaxInactivityDurationInitialDelay** (long long value)  
*Sets the MaxInactivityDurationInitialDelay setting.*

### Protected Member Functions

- **commands::DataStructure \* doUnmarshal** (decaf::io::DataInputStream \*dis)  
throw ( decaf::io::IOException )  
*Perform the actual unmarshal of data from the given DataInputStream return the unmarshalled DataStrucutre object once done, caller takes ownership of this object.*
- void **destroyMarshallers** ()  
*Cleans up all registered Marshallers and empties the dataMarshallers vector.*

### Static Protected Attributes

- static const unsigned char **NULL\_TYPE**
- static const int **DEFAULT\_VERSION** = 1

### 6.598.1 Constructor & Destructor Documentation

- 6.598.1.1 **activemq::wireformat::openwire::OpenWireFormat::OpenWireFormat** ( const decaf::util::Properties & *properties* )

Constructs a new **OpenWireFormat** (p. 2971) object.

#### Parameters

<i>properties</i>	- can contain optional config params.
-------------------	---------------------------------------

6.598.1.2 `virtual activemq::wireformat::openwire::OpenWireFormat::~~OpenWireFormat ( )`  
[virtual]

## 6.598.2 Member Function Documentation

6.598.2.1 `void activemq::wireformat::openwire::OpenWireFormat::addMarshaller (`  
`marshal::DataStreamMarshaller * marshaller )`

Allows an external source to add marshalers to this object for types that may be marshaled or unmarshaled.

### Parameters

<i>marshaller</i>	- the Marshaler to add to the collection.
-------------------	---

6.598.2.2 `virtual Pointer<transport::Transport>`  
`activemq::wireformat::openwire::OpenWireFormat::createNegotiator (`  
`const Pointer< transport::Transport > & transport ) throw (`  
`decaf::lang::exceptions::UnsupportedOperationException )`  
[virtual]

If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

### Returns

new instance of a **WireFormatNegotiator** (p. 4129).

Implements **activemq::wireformat::WireFormat** (p. 4089).

6.598.2.3 `void activemq::wireformat::openwire::OpenWireFormat::destroyMarshalers ( )`  
[protected]

Cleans up all registered Marshallers and empties the dataMarshallers vector.

This should be called before a reconfiguration of the version marshalers, or on destruction of this object

6.598.2.4 `commands::DataStructure* ac-`  
`tivemq::wireformat::openwire::OpenWireFormat::doUnmarshal (`  
`decaf::io::DataInputStream * dis ) throw ( decaf::io::IOException )`  
[protected]

Perform the actual unmarshal of data from the given DataInputStream return the unmarshalled DataStrucutre object once done, caller takes ownership of this object.

This method can return null if the type of the object to unmarshal is NULL, empty data.

## 6.598 activemq::wireformat::openwire::OpenWireFormat Class Reference 2979

### Parameters

<i>dis</i>	- DataInputStream to read from
------------	--------------------------------

### Returns

new DataStructure\* that the caller owns

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal
--------------------	---

**6.598.2.5** `int activemq::wireformat::openwire::OpenWireFormat::getCacheSize ( ) const`  
[inline]

Returns the currently set Cache size.

### Returns

the current value of the broker's cache size.

**6.598.2.6** `long long activemq::wireformat::openwire::OpenWireFormat::getMaxInactivityDuration`  
`( ) const` [inline]

Gets the MaxInactivityDuration setting.

### Returns

maximum inactivity duration value in milliseconds.

**6.598.2.7** `long long activemq::wireformat::openwire::OpenWireFormat::getMaxInactivityDurationInitialDelay`  
`( ) const` [inline]

Gets the MaxInactivityDurationInitialDelay setting.

### Returns

maximum inactivity duration initial delay value in milliseconds.

**6.598.2.8** `virtual const Pointer<commands::WireFormatInfo>&`  
`activemq::wireformat::openwire::OpenWireFormat::getPreferredWireFormatInfo ( )`  
`const` [inline, virtual]

Gets the Preferred WireFormatInfo object that this class holds.

### Returns

pointer to a preferred WireFormatInfo object

**6.598.2.9** `int activemq::wireformat::openwire::OpenWireFormat::getVersion ( ) const`  
`[inline, virtual]`

Get the current Wireformat Version.

#### Returns

int that identifies the version

Implements **activemq::wireformat::WireFormat** (p. 4090).

**6.598.2.10** `virtual bool activemq::wireformat::openwire::OpenWireFormat::hasNegotiator ( )`  
`const [inline, virtual]`

Returns true if this **WireFormat** (p.4088) has a Negotiator that needs to wrap the Transport that uses it.

#### Returns

true if the **WireFormat** (p. 4088) provides a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 4090).

**6.598.2.11** `virtual bool activemq::wireformat::openwire::OpenWireFormat::inReceive ( ) const`  
`[inline, virtual]`

Is there a Message being unmarshaled?

#### Returns

true while in the doUnmarshal method.

Implements **activemq::wireformat::WireFormat** (p. 4090).

**6.598.2.12** `bool activemq::wireformat::openwire::OpenWireFormat::isCacheEnabled ( ) const`  
`[inline]`

Checks if the cacheEnabled flag is on.

#### Returns

true if the flag is on.

**6.598.2.13** `bool activemq::wireformat::openwire::OpenWireFormat::isSizePrefixDisabled ( )`  
`const [inline]`

Checks if the sizePrefixDisabled flag is on.

## 6.598 activemq::wireformat::openwire::OpenWireFormat Class Reference 2981

### Returns

true if the flag is on.

**6.598.2.14** `bool activemq::wireformat::openwire::OpenWireFormat::isStackTraceEnabled ( )`  
`const [inline]`

Checks if the stackTraceEnabled flag is on.

### Returns

true if the flag is on.

**6.598.2.15** `bool activemq::wireformat::openwire::OpenWireFormat::isTcpNoDelayEnabled ( )`  
`const [inline]`

Checks if the tcpNoDelayEnabled flag is on.

### Returns

true if the flag is on.

**6.598.2.16** `bool activemq::wireformat::openwire::OpenWireFormat::isTightEncodingEnabled ( )`  
`const [inline]`

Checks if the tightEncodingEnabled flag is on.

### Returns

true if the flag is on.

**6.598.2.17** `void activemq::wireformat::openwire::OpenWireFormat::looseMarshalNestedObject (`  
`commands::DataStructure * o, decaf::io::DataOutputStream * dataOut`  
`) throw ( decaf::io::IOException )`

Utility method to loosely Marshal an object that is derived from the DataStructure interface.

The marshaled data is written to the passed in DataOutputStream.

### Parameters

<i>o</i>	- DataStructure derived Object to Marshal
<i>dataOut</i>	- DataOutputStream to write the data to

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.598.2.18** **commands::DataStructure\*** **activemq::wireformat::openwire::OpenWireFormat::looseUnmarshalNestedObject (**  
**decaf::io::DataInputStream \* *dis* ) throw ( decaf::io::IOException )**

Utility method to unmarshal an DataStructure object from an DataInputStream using the Loose Unmarshaling format.

Will read the Data and construct a new DataStructure based Object, the pointer to the Object returned is now owned by the caller.

**Parameters**

<i>dis</i>	- the DataInputStream to read the data from
------------	---

**Returns**

a new DataStructure derived Object pointer

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.598.2.19** **virtual void** **activemq::wireformat::openwire::OpenWireFormat::marshal**  
**( const Pointer< commands::Command > & *command*,**  
**const activemq::transport::Transport \* *transport*,**  
**decaf::io::DataOutputStream \* *out* ) throw ( decaf::io::IOException )**  
 [virtual]

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

**Parameters**

<i>command</i>	The Command to Marshal.
<i>transport</i>	The Transport instance that called this method.
<i>out</i>	The output stream to write the command to.

**Exceptions**

<i>IOException</i>
--------------------

Implements **activemq::wireformat::WireFormat** (p. 4091).

## **6.598 activemq::wireformat::openwire::OpenWireFormat Class Reference 2983**

**6.598.2.20** void activemq::wireformat::openwire::OpenWireFormat::renegotiateWireFormat  
( const commands::WireFormatInfo & *info* ) throw (  
decaf::lang::exceptions::IllegalStateException )

Called to re-negotiate the settings for the WireFormatInfo, these determine how the client and broker communicate.

### **Parameters**

<i>info</i>	- The new Wireformat Info settings
-------------	------------------------------------

### **Exceptions**

<i>IllegalStateException</i>	is wire format can't be negotiated.
------------------------------	-------------------------------------

**6.598.2.21** void activemq::wireformat::openwire::OpenWireFormat::setCacheEnabled ( bool  
*cacheEnabled* ) [inline]

Sets if the cacheEnabled flag is on.

### **Parameters**

<i>cacheEnabled</i>	- true to turn flag is on
---------------------	---------------------------

**6.598.2.22** void activemq::wireformat::openwire::OpenWireFormat::setCacheSize ( int *value* )  
[inline]

Sets the current Cache size.

### **Parameters**

<i>value</i>	- the value to send as the broker's cache size.
--------------	---

**6.598.2.23** void activemq::wireformat::openwire::OpenWireFormat::setMaxInactivityDuration ( long long *value* ) [inline]

Sets the MaxInactivityDuration setting.

### **Parameters**

<i>value</i>	- the Max inactivity duration value in milliseconds.
--------------	--

**6.598.2.24** `void activemq::wireformat::openwire::OpenWireFormat::setMaxInactivityDurationInitialDelay ( long long value ) [inline]`

Sets the MaxInactivityDurationInitialDelay setting.

#### Parameters

<i>value</i>	- the Max inactivity Initial Delay duration value in milliseconds.
--------------	--

**6.598.2.25** `virtual void activemq::wireformat::openwire::OpenWireFormat::setPreferedWireFormatInfo ( const Pointer< commands::WireFormatInfo > & info ) throw ( decaf::lang::exceptions::IllegalStateException ) [virtual]`

Configures this object using the provided WireformatInfo object.

#### Parameters

<i>info</i>	- a WireFormatInfo object, takes ownership.
-------------	---

**6.598.2.26** `void activemq::wireformat::openwire::OpenWireFormat::setSizePrefixDisabled ( bool sizePrefixDisabled ) [inline]`

Sets if the sizePrefixDisabled flag is on.

#### Parameters

<i>sizePrefixDisabled</i>	- true to turn flag is on
---------------------------	---------------------------

**6.598.2.27** `void activemq::wireformat::openwire::OpenWireFormat::setStackTraceEnabled ( bool stackTraceEnabled ) [inline]`

Sets if the stackTraceEnabled flag is on.

#### Parameters

<i>stack-TraceEnabled</i>	- true to turn flag is on
---------------------------	---------------------------

**6.598.2.28** `void activemq::wireformat::openwire::OpenWireFormat::setTcpNoDelayEnabled ( bool tcpNoDelayEnabled ) [inline]`

Sets if the tcpNoDelayEnabled flag is on.



## 6.598 activemq::wireformat::openwire::OpenWireFormat Class Reference 2985

### Parameters

<i>tcpNoDelayEnabled</i>	- true to turn flag is on
--------------------------	---------------------------

**6.598.2.29** void activemq::wireformat::openwire::OpenWireFormat::setTightEncodingEnabled ( bool *tightEncodingEnabled* ) [inline]

Sets if the tightEncodingEnabled flag is on.

### Parameters

<i>tightEncodingEnabled</i>	- true to turn flag is on
-----------------------------	---------------------------

**6.598.2.30** void activemq::wireformat::openwire::OpenWireFormat::setVersion ( int *version* ) throw ( decaf::lang::exceptions::IllegalArgumentException ) [virtual]

Set the current Wireformat Version.

### Parameters

<i>version</i>	- int that identifies the version
----------------	-----------------------------------

Implements **activemq::wireformat::WireFormat** (p. 4091).

**6.598.2.31** virtual int activemq::wireformat::openwire::OpenWireFormat::tightMarshalNestedObject1 ( commands::DataStructure \* *object*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Utility method for Tight Marshaling the given object to the boolean stream passed.

### Parameters

<i>object</i>	- The DataStructure to marshal
<i>bs</i>	- the BooleanStream to write to

### Returns

size of the data returned.

**6.598.2.32** `void activemq::wireformat::openwire::OpenWireFormat::tightMarshalNestedObject2 ( commands::DataStructure * o, decaf::io::DataOutputStream * ds, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`

Utility method that will Tight marshal some internally nested object that implements the DataStructure interface.

Writes the data to the Data Output Stream provided.

#### Parameters

<i>o</i>	- DataStructure object
<i>ds</i>	- DataOuputStream for writing
<i>bs</i>	- BooleanStream

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.598.2.33** `commands::DataStructure* activemq::wireformat::openwire::OpenWireFormat::tightUnmarshalNestedObject ( decaf::io::DataInputStream * dis, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`

Utility method used to Unmarshal a Nested DataStructure type object from the given DataInputStream.

The DataStructure instance that is returned is now the property of the caller.

#### Parameters

<i>dis</i>	- DataInputStream to read from
<i>bs</i>	- BooleanStream to read from

#### Returns

Newly allocated DataStructure Object

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

**6.598.2.34** `virtual Pointer<commands::Command> activemq::wireformat::openwire::OpenWireFormat::unmarshal ( const activemq::transport::Transport * transport, decaf::io::DataInputStream * in ) throw ( decaf::io::IOException )`  
[virtual]

Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and un-marshaled into the correct form.

## 6.599 activemq::wireformat::openwire::OpenWireFormatFactory Class Reference

2987

Returns a Pointer to the newly un-marshaled Command.

### Parameters

<i>transport</i>	- Pointer to the transport that is making this request.
<i>in</i>	- the input stream to read the command from.

### Returns

the newly marshaled Command, caller owns the pointer

### Exceptions

<i>IOException</i>	
--------------------	--

Implements **activemq::wireformat::WireFormat** (p. 4091).

### 6.598.3 Field Documentation

**6.598.3.1** `const int activemq::wireformat::openwire::OpenWireFormat::DEFAULT_VERSION = 1` [static, protected]

**6.598.3.2** `const unsigned char activemq::wireformat::openwire::OpenWireFormat::NULL_TYPE` [static, protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/OpenWireFormat.h`

## 6.599 activemq::wireformat::openwire::OpenWireFormatFactory Class Reference

```
#include <src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h>
```

Inheritance diagram for `activemq::wireformat::openwire::OpenWireFormatFactory`:

### Public Member Functions

- **OpenWireFormatFactory ()**  
*Constructor - Sets Defaults for all properties, these are all subject to change once the createWireFormat method is called.*
- `virtual ~OpenWireFormatFactory ()`
- `virtual Pointer< wireformat::WireFormat > createWireFormat (const decaf::util::Properties &properties) throw ( decaf::lang::exceptions::IllegalStateException )`

*Creates a new **WireFormat** (p. 4088) Object passing it a set of properties from which it can obtain any optional settings.*

### 6.599.1 Constructor & Destructor Documentation

6.599.1.1 `activemq::wireformat::openwire::OpenWireFormatFactory::OpenWireFormatFactory ( ) [inline]`

Constructor - Sets Defaults for all properties, these are all subject to change once the `createWireFormat` method is called.

URL options ----- `wireFormat.stackTraceEnabled` `wireFormat.cacheEnabled`  
`wireFormat.tcpNoDelayEnabled` `wireFormat.tightEncodingEnabled` `wireFormat.sizePrefixDisabled`  
`wireFormat.maxInactivityDuration` `wireFormat.maxInactivityDurationInitialDelay`

6.599.1.2 `virtual activemq::wireformat::openwire::OpenWireFormatFactory::~~OpenWireFormatFactory ( ) [inline, virtual]`

### 6.599.2 Member Function Documentation

6.599.2.1 `virtual Pointer<wireformat::WireFormat> activemq::wireformat::openwire::OpenWireFormatFactory::createWireFormat ( const decaf::util::Properties & properties ) throw ( decaf::lang::exceptions::IllegalStateException ) [virtual]`

Creates a new **WireFormat** (p. 4088) Object passing it a set of properties from which it can obtain any optional settings.

#### Parameters

<i>properties</i>	- the Properties for this <b>WireFormat</b> (p. 4088)
-------------------	---

Implements `activemq::wireformat::WireFormatFactory` (p. 4093).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h`

## 6.600 `activemq::wireformat::openwire::OpenWireFormatNegotiator` Class Reference

```
#include <src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h>
```

Inheritance diagram for `activemq::wireformat::openwire::OpenWireFormatNegotiator`:

## Public Member Functions

- **OpenWireFormatNegotiator** (**OpenWireFormat** \*wireFormat, const **Pointer**< **transport::Transport** > &next)  
*Constructor - Initializes this object around another Transport.*
- virtual ~**OpenWireFormatNegotiator** ()
- virtual void **oneway** (const **Pointer**< **commands::Command** > &command)  
throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends a one-way command.*
- virtual **Pointer**< **commands::Response** > **request** (const **Pointer**< **commands::Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends the given request to the server and waits for the response.*
- virtual **Pointer**< **commands::Response** > **request** (const **Pointer**< **commands::Command** > &command, unsigned int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends the given request to the server and waits for the response.*
- virtual void **onCommand** (const **Pointer**< **commands::Command** > &command)  
*This is called in the context of the nested transport's reading thread.*
- virtual void **onTransportException** (**transport::Transport** \*source, const **decaf::lang::Exception** &ex)  
*Event handler for an exception from a command transport.*
- virtual void **start** () throw ( decaf::io::IOException )  
*Starts this transport object and creates the thread for polling on the input stream for commands.*
- virtual void **close** () throw ( decaf::io::IOException )  
*Stops the polling thread and closes the streams.*

### 6.600.1 Constructor & Destructor Documentation

#### 6.600.1.1 activemq::wireformat::openwire::OpenWireFormatNegotiator::OpenWireFormatNegotiator ( **OpenWireFormat** \* wireFormat, const **Pointer**< **transport::Transport** > & next )

Constructor - Initializes this object around another Transport.

## Parameters

<i>wireFormat</i>	- The <b>WireFormat</b> (p. 4088) object we use to negotiate
<i>next</i>	- The next transport in the chain

**6.600.1.2** `virtual activemq::wireformat::openwire::OpenWireFormatNegotiator::~~OpenWireFormatNegotiator ( ) [virtual]`

## 6.600.2 Member Function Documentation

**6.600.2.1** `virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::close ( ) throw ( decaf::io::IOException ) [virtual]`

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

### Exceptions

<i>IOException</i> if errors occur.
-------------------------------------

Reimplemented from `activemq::transport::TransportFilter` (p. 4007).

**6.600.2.2** `virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::onCommand ( const Pointer< commands::Command > & command ) [virtual]`

This is called in the context of the nested transport's reading thread.

In the case of a response object, updates the request map and notifies those waiting on the response. Non-response messages are just delegated to the command listener.

### Parameters

<i>command</i>	the received from the nested transport.
----------------	---

Reimplemented from `activemq::transport::TransportFilter` (p. 4009).

**6.600.2.3** `virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::oneway ( const Pointer< commands::Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Sends a one-way command.

Does not wait for any response from the broker. First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

## Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

## Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 4010).

6.600.2.4 **virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::onTransportException**  
( **transport::Transport \* source**, **const decaf::lang::Exception & ex** )  
[virtual]

Event handler for an exception from a command transport.

## Parameters

<i>source</i>	The source of the exception
<i>ex</i>	The exception.

6.600.2.5 **virtual Pointer<commands::Response>**  
**activemq::wireformat::openwire::OpenWireFormatNegotiator::request**  
( **const Pointer< commands::Command >**  
**& command** ) **throw ( decaf::io::IOException,**  
**decaf::lang::exceptions::UnsupportedOperationException )**  
[virtual]

Sends the given request to the server and waits for the response.

First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

## Parameters

<i>command</i>	The request to send.
----------------	----------------------

## Returns

the response from the server.

## Exceptions

<i>IOException</i>	if an error occurs with the request.
--------------------	--------------------------------------

Reimplemented from **activemq::transport::TransportFilter** (p. 4011).

```

6.600.2.6  virtual Pointer<commands::Response>
            activemq::wireformat::openwire::OpenWireFormatNegotiator::request
            ( const Pointer< commands::Command > & command,
              unsigned int timeout ) throw ( decaf::io::IOException,
                                              decaf::lang::exceptions::UnsupportedOperationException )
            [virtual]

```

Sends the given request to the server and waits for the response.

First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

#### Parameters

<i>command</i>	The request to send.
<i>timeout</i>	The time to wait for the response.

#### Returns

the response from the server.

#### Exceptions

<i>IOException</i>	if an error occurs with the request.
--------------------	--------------------------------------

Reimplemented from **activemq::transport::TransportFilter** (p. 4011).

```

6.600.2.7  virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::start ( )
            throw ( decaf::io::IOException ) [virtual]

```

Starts this transport object and creates the thread for polling on the input stream for commands.

If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

#### Exceptions

<i>IOException</i>	if an error occurs or if this transport has already been closed.
--------------------	--

Reimplemented from **activemq::transport::TransportFilter** (p. 4012).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/**OpenWireFormatNegotiator.h**

## 6.601 activemq::wireformat::openwire::OpenWireResponseBuilder Class Reference

```
#include <src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h>
```



Inheritance diagram for activemq::wireformat::openwire::OpenWireResponseBuilder:

## Public Member Functions

- **OpenWireResponseBuilder** ()
- virtual **~OpenWireResponseBuilder** ()
- virtual **Pointer< commands::Response > buildResponse** (const **Pointer< commands::Command > &command**)

*Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.*

- virtual void **buildIncomingCommands** (const **Pointer< commands::Command > &command**, **decaf::util::StlQueue< Pointer< commands::Command > > &queue**)

*When called the ResponseBuilder must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.*

## 6.601.1 Constructor & Destructor Documentation

6.601.1.1 **activemq::wireformat::openwire::OpenWireResponseBuilder::OpenWireResponseBuilder** ( ) [inline]

6.601.1.2 **virtual activemq::wireformat::openwire::OpenWireResponseBuilder::~~OpenWireResponseBuilder** ( ) [inline, virtual]

## 6.601.2 Member Function Documentation

6.601.2.1 **virtual void activemq::wireformat::openwire::OpenWireResponseBuilder::buildIncomingCommands** ( const **Pointer< commands::Command > &command**, **decaf::util::StlQueue< Pointer< commands::Command > > &queue** ) [virtual]

When called the ResponseBuilder must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.

### Parameters

<i>command</i>	- The Command being sent to the Broker.
<i>queue</i>	- Queue of Command sent back from the broker.

Implements **activemq::transport::mock::ResponseBuilder** (p. 3383).

**6.601.2.2** `virtual Pointer<commands::Response>`  
`activemq::wireformat::openwire::OpenWireResponseBuilder::buildResponse ( const`  
`Pointer< commands::Command > & command ) [virtual]`

Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.

#### Parameters

<i>command</i>	- The command to build a response for
----------------	---------------------------------------

#### Returns

A Response object pointer, or NULL if no response.

Implements `activemq::transport::mock::ResponseBuilder` (p. 3384).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h`

## 6.602 decaf::io::OutputStream Class Reference

Base interface for any class that wants to represent an output stream of bytes.

```
#include <src/main/decaf/io/OutputStream.h>
```

Inheritance diagram for `decaf::io::OutputStream`:

#### Public Member Functions

- **OutputStream** ()
- virtual **~OutputStream** ()
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes this object and deallocates the appropriate resources.  
The object is generally no longer usable after calling close.*

##### Exceptions

<b>IOException</b> (p. 2210)	if an error occurs while closing.
------------------------------	-----------------------------------

- virtual void **flush** () throw ( decaf::io::IOException )  
*Flushes this stream by writing any buffered output to the underlying stream.*

##### Exceptions

<b>IOException</b> (p. 2210)	if an I/O error occurs.
------------------------------	-------------------------

- virtual void **write** (unsigned char c) throw ( decaf::io::IOException )  
*Writes a single byte to the output stream.*
- virtual void **write** (const unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes an array of bytes to the output stream.*
- virtual void **write** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes an array of bytes to the output stream in order starting at buffer[offset] and proceeding until the number of bytes specified by the length argument are written or an error occurs.*
- virtual std::string **toString** () const  
*Output a String representation of this object.*
- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Locks the object.*
- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Attempts to Lock the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Unlocks the object.*
- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals a waiter on this object that it can now wake up and continue.*
- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals the waiters on this object that it can now wake up and continue.*

## Protected Member Functions

- virtual void **doWriteByte** (unsigned char value)=0 throw ( decaf::io::IOException )
- virtual void **doWriteArray** (const unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### 6.602.1 Detailed Description

Base interface for any class that wants to represent an output stream of bytes.

Since

1.0

### 6.602.2 Constructor & Destructor Documentation

6.602.2.1 decaf::io::OutputStream::OutputStream ( )

6.602.2.2 virtual decaf::io::OutputStream::~~OutputStream ( ) [virtual]

### 6.602.3 Member Function Documentation

6.602.3.1 virtual void decaf::io::OutputStream::close ( ) throw ( decaf::io::IOException )  
[virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

#### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an error occurs while closing.
--	-----------------------------------

The default implementation of this method does nothing.

Implements **decaf::io::Closeable** (p. 1181).

Reimplemented in **decaf::internal::io::StandardErrorOutputStream** (p. 3687), **decaf::internal::io::StandardOutputStream** (p. 3690), **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream** (p. 2970), **decaf::internal::net::tcp::TcpSocketOutputStream** (p. 3864), **decaf::io::FilterOutputStream**

(p. 1959), and **decaf::util::zip::DeflaterOutputStream** (p. 1772).

**6.602.3.2** `virtual void decaf::io::OutputStream::doWriteArray ( const unsigned char * buffer, int size ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`  
[protected, virtual]

Reimplemented in **decaf::io::BufferedOutputStream** (p. 950), and **decaf::io::FilterOutputStream** (p. 1960).

**6.602.3.3** `virtual void decaf::io::OutputStream::doWriteArrayBounded ( const unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`  
[protected, virtual]

Reimplemented in **activemq::io::LoggingOutputStream** (p. 2477), **decaf::internal::io::StandardErrorOutputStream** (p. 3687), **decaf::internal::io::StandardOutputStream** (p. 3691), **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream** (p. 2971), **decaf::internal::net::tcp::TcpSocketOutputStream** (p. 3865), **decaf::io::BufferedOutputStream** (p. 950), **decaf::io::ByteArrayOutputStream** (p. 1048), **decaf::io::DataOutputStream** (p. 1630), **decaf::io::FilterOutputStream** (p. 1960), **decaf::util::zip::CheckedOutputStream** (p. 1173), and **decaf::util::zip::DeflaterOutputStream** (p. 1773).

**6.602.3.4** `virtual void decaf::io::OutputStream::doWriteByte ( unsigned char value ) throw ( decaf::io::IOException )` [protected, pure virtual]

Implemented in **activemq::io::LoggingOutputStream** (p. 2477), **decaf::internal::io::StandardErrorOutputStream** (p. 3688), **decaf::internal::io::StandardOutputStream** (p. 3691), **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream** (p. 2971), **decaf::internal::net::tcp::TcpSocketOutputStream** (p. 3865), **decaf::io::BufferedOutputStream** (p. 950), **decaf::io::ByteArrayOutputStream** (p. 1048), **decaf::io::DataOutputStream** (p. 1630), **decaf::io::FilterOutputStream** (p. 1960), **decaf::util::zip::CheckedOutputStream** (p. 1173), and **decaf::util::zip::DeflaterOutputStream** (p. 1773).

**6.602.3.5** `virtual void decaf::io::OutputStream::flush ( ) throw ( decaf::io::IOException )`  
[virtual]

Flushes this stream by writing any buffered output to the underlying stream.

### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
--	-------------------------

The default implementation of this method does nothing.

Implements **decaf::io::Flushable** (p. 1998).

Reimplemented in **decaf::internal::io::StandardErrorOutputStream** (p. 3688), **decaf::internal::io::StandardOutputStream** (p. 3691), **decaf::io::BufferedOutputStream** (p. 951), and **decaf::io::FilterOutputStream** (p. 1960).

**6.602.3.6** `virtual void decaf::io::OutputStream::lock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Locks the object.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3812).

**6.602.3.7** `virtual void decaf::io::OutputStream::notify ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException ) [inline, virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

#### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3813).

**6.602.3.8** `virtual void decaf::io::OutputStream::notifyAll ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException ) [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

#### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3814).

**6.602.3.9** virtual std::string decaf::io::OutputStream::toString ( ) const [virtual]

Output a String representation of this object.

The default version of this method just prints the Class Name.

#### Returns

a string representation of the object.

Reimplemented in **decaf::io::ByteArrayOutputStream** (p. 1049), and **decaf::io::FilterOutputStream** (p. 1961).

**6.602.3.10** virtual bool decaf::io::OutputStream::tryLock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

#### Returns

true if the lock was acquired, false if it is already held by another thread.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3815).

**6.602.3.11** virtual void decaf::io::OutputStream::unlock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]

Unlocks the object.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3816).

**6.602.3.12** virtual void decaf::io::OutputStream::wait ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

**Exceptions**

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3818).

**6.602.3.13** `virtual void decaf::io::OutputStream::wait ( long long millisecs ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

**Parameters**

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

**Exceptions**

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3819).

**6.602.3.14** `virtual void decaf::io::OutputStream::wait ( long long millisecs, int nanos ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.



**Parameters**

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

**Exceptions**

<i>IllegalArgumentException</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3820).

**6.602.3.15** virtual void decaf::io::OutputStream::write ( unsigned char *c* ) throw ( decaf::io::IOException ) [virtual]

Writes a single byte to the output stream.

The default implementation of this method calls the pure virtual method doWriteByte which must be implemented by any subclass of the **OutputStream** (p. 2991).

**Parameters**

<i>c</i>	The byte to write to the sink.
----------	--------------------------------

**Exceptions**

<i>IOException</i> (p. 2210)	if an I/O error occurs.
---------------------------------	-------------------------

**6.602.3.16** virtual void decaf::io::OutputStream::write ( const unsigned char \* *buffer*, int *size* ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Writes an array of bytes to the output stream.

The entire contents of the given vector are written to the output stream.

The default implementation of this method simply calls the doWriteArray which writes the contents of the array using the doWriteByte method repeatedly. It is recommended that a subclass override doWriteArray to provide more performant means of writing the array.

**Parameters**

<i>buffer</i>	The vector of bytes to write.
<i>size</i>	The size of the buffer passed.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
<i>NullPointerException</i>	thrown if buffer is Null.
<i>IndexOutOfBoundsException</i>	if size value is negative.

**6.602.3.17** `virtual void decaf::io::OutputStream::write ( const unsigned char * buffer,  
int size, int offset, int length ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Writes an array of bytes to the output stream in order starting at buffer[offset] and proceeding until the number of bytes specified by the length argument are written or an error occurs.

The default implementation of this method simply calls the doWriteArrayBounded method which writes the contents of the array using the doWriteByte method repeatedly. It is recommended that a subclass override doWriteArrayBounded to provide more performant means of writing the array.

**Parameters**

<i>buffer</i>	The array of bytes to write.
<i>size</i>	The size of the buffer array passed.
<i>offset</i>	The position to start writing in buffer.
<i>length</i>	The number of bytes from the buffer to be written.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
<i>NullPointerException</i>	thrown if buffer is Null.
<i>IndexOutOfBoundsException</i>	if the offset + length > size. or one of the parameters is negative.

The documentation for this class was generated from the following file:

- src/main/decaf/io/OutputStream.h

**6.603 decaf::io::OutputStreamWriter Class Reference**

A class for turning a character stream into a byte stream.

```
#include <src/main/decaf/io/OutputStreamWriter.h>
```

Inheritance diagram for decaf::io::OutputStreamWriter:

### Public Member Functions

- **OutputStreamWriter** (**OutputStream** \*stream, bool own=false)  
*Creates a new **OutputStreamWriter** (p. 2999).*
- virtual ~**OutputStreamWriter** ()
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes this object and deallocates the appropriate resources.*
- virtual void **flush** () throw ( decaf::io::IOException )  
*Flushes this stream by writing any buffered output to the underlying stream.*

### Protected Member Functions

- virtual void **doWriteArrayBounded** (const char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Override this method to customize the functionality of the method write( char\* buffer, int size, int offset, int length ).*
- virtual void **checkClosed** () const throw ( decaf::io::IOException )

#### 6.603.1 Detailed Description

A class for turning a character stream into a byte stream.

See also

**InputStreamReader** (p. 2116)

Since

1.0

#### 6.603.2 Constructor & Destructor Documentation

**6.603.2.1** decaf::io::OutputStreamWriter::OutputStreamWriter ( **OutputStream** \* stream, bool own = false )

Creates a new **OutputStreamWriter** (p. 2999).

### Parameters

<i>stream</i>	The <b>OutputStream</b> (p. 2991) to wrap. (cannot be NULL).
<i>own</i>	Indicates whether this instance own the given <b>OutputStream</b> (p. 2991). If true then the <b>OutputStream</b> (p. 2991) is destroyed when this class is.

### Exceptions

<i>NullPointerException</i>	if the stream is NULL.
-----------------------------	------------------------

6.603.2.2 virtual **decaf::io::OutputStreamWriter::~~OutputStreamWriter** ( ) [virtual]

### 6.603.3 Member Function Documentation

6.603.3.1 virtual void **decaf::io::OutputStreamWriter::checkClosed** ( ) const throw ( **decaf::io::IOException** ) [protected, virtual]

6.603.3.2 virtual void **decaf::io::OutputStreamWriter::close** ( ) throw ( **decaf::io::IOException** ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

### Exceptions

<i>IOException</i> (p. 2210)	if an error occurs while closing.
---------------------------------	-----------------------------------

Implements **decaf::io::Closeable** (p. 1181).

6.603.3.3 virtual void **decaf::io::OutputStreamWriter::doWriteArrayBounded** ( const char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( **decaf::io::IOException**, **decaf::lang::exceptions::NullPointerException**, **decaf::lang::exceptions::IndexOutOfBoundsException** ) [protected, virtual]

Override this method to customize the functionality of the method write( char\* buffer, int size, int offset, int length ).

All subclasses must override this method to provide the basic **Writer** (p. 4133) functionality.

Implements **decaf::io::Writer** (p. 4137).

6.603.3.4 virtual void **decaf::io::OutputStreamWriter::flush** ( ) throw ( **decaf::io::IOException** ) [virtual]

Flushes this stream by writing any buffered output to the underlying stream.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
--	-------------------------

Implements **decaf::io::Flushable** (p. 1998).

The documentation for this class was generated from the following file:

- src/main/decaf/io/OutputStreamWriter.h

**6.604 activemq::commands::PartialCommand Class Reference**

```
#include <src/main/activemq/commands/PartialCommand.h>
```

Inheritance diagram for activemq::commands::PartialCommand:

**Public Member Functions**

- **PartialCommand** ()
- virtual **~PartialCommand** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **PartialCommand \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual int **getCommandId** () const
- virtual void **setCommandId** (int commandId)
- virtual const std::vector< unsigned char > & **getData** () const
- virtual std::vector< unsigned char > & **getData** ()
- virtual void **setData** (const std::vector< unsigned char > &data)

**Static Public Attributes**

- static const unsigned char **ID\_PARTIALCOMMAND** = 60

**Protected Attributes**

- int **commandId**
- std::vector< unsigned char > **data**

**6.604.1 Constructor & Destructor Documentation**

6.604.1.1 **activemq::commands::PartialCommand::PartialCommand ( )**

6.604.1.2 **virtual activemq::commands::PartialCommand::~~PartialCommand ( )**  
[virtual]

**6.604.2 Member Function Documentation**

6.604.2.1 **virtual PartialCommand\* activemq::commands::PartialCommand::cloneDataStructure ( ) const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

**Returns**

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 2372).

6.604.2.2 **virtual void activemq::commands::PartialCommand::copyDataStructure ( const DataStructure \* *src* )** [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

**Parameters**

<i>src</i>	- Source Object
------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1715).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 2372).

**6.604.2.3** `virtual bool activemq::commands::PartialCommand::equals ( const DataStructure * value ) const` [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1716).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 2373).

**6.604.2.4** `virtual int activemq::commands::PartialCommand::getCommandId ( ) const` [virtual]

**6.604.2.5** `virtual std::vector<unsigned char>& activemq::commands::PartialCommand::getData ( )` [virtual]

**6.604.2.6** `virtual const std::vector<unsigned char>& activemq::commands::PartialCommand::getData ( ) const` [virtual]

**6.604.2.7** `virtual unsigned char activemq::commands::PartialCommand::getDataStructureType ( ) const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 2373).

**6.604.2.8** `virtual void activemq::commands::PartialCommand::setCommandId ( int commandId ) [virtual]`

**6.604.2.9** `virtual void activemq::commands::PartialCommand::setData ( const std::vector< unsigned char > & data ) [virtual]`

**6.604.2.10** `virtual std::string activemq::commands::PartialCommand::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 841).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 2373).

## 6.604.3 Field Documentation

**6.604.3.1** `int activemq::commands::PartialCommand::commandId [protected]`

**6.604.3.2** `std::vector<unsigned char> activemq::commands::PartialCommand::data [protected]`

**6.604.3.3** `const unsigned char activemq::commands::PartialCommand::ID_PARTIALCOMMAND = 60 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/PartialCommand.h`

## 6.605 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3005).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/PartialCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller`:



- **PartialCommandMarshaller ()**
- virtual **~PartialCommandMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.605.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3005).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.605.2 Constructor & Destructor Documentation

6.605.2.1 `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::PartialCommandMarshaller ( ) [inline]`

6.605.2.2 `virtual activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::~~PartialCommandMarshaller ( ) [inline, virtual]`

## 6.605.3 Member Function Documentation

6.605.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller` (p. 2375).

6.605.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller` (p. 2375).

6.605.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides the data sink

## 6.605

**activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller**

**Class Reference**

**3011**

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2375).

**6.605.3.4** virtual void **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2376).

**6.605.3.5** virtual int **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::tightMarshal1** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2376).

**6.605.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2377).

**6.605.3.7** `virtual void activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

6.606

**activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller**

**Class Reference**

**3013**

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2377).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**PartialCommandMarshaller.h**

## 6.606 **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3010).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller**:

### Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.606.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3010).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.606.2 Constructor & Destructor Documentation

6.606.2.1 **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::PartialCommandMarshaller**  
( ) [inline]

6.606.2.2 **virtual activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::~~PartialCommandMarshaller**  
( ) [inline, virtual]

### 6.606.3 Member Function Documentation

6.606.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::createObject** ( ) **const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2383).

6.606.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

6.606

**activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller**

**Class Reference**

**3015**

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2384).

**6.606.3.3** virtual void **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::looseMarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2384).

**6.606.3.4** virtual void **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller` (p. 2384).

```
6.606.3.5  virtual int activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1690).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller` (p. 2385).

```
6.606.3.6  virtual void activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



## 6.607

**activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller**

### Class Reference

3017

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2385).

```
6.606.3.7 virtual void activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2386).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**PartialCommandMarshaller.h**

## 6.607 **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3014).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller**:

### Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()

- virtual **commands::DataStructure \* createObject ()** const

*Creates a new instance of this marshalable type.*

- virtual unsigned char **getDataStructureType ()** const

*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)**  
throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn)** throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut)** throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.607.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3014).  
NOTE!: This file is auto generated - do not modify! if you need to make a change,  
please see the Java Classes in the activemq-openwire-generator module

6.607.2.1 `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::PartialCommandMarshaller ( ) [inline]`

6.607.2.2 `virtual activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::~~PartialCommandMarshaller ( ) [inline, virtual]`

6.607.3 Member Function Documentation

6.607.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 2388).

6.607.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 2388).

6.607.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- Binary Writer that provides a data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 2388).

**6.607.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 2389).

**6.607.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

6.607

**activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller**

**Class Reference**

**3021**

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 2389).

**6.607.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 2390).

**6.607.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 2390).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h`

## 6.608 **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3019).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller**:

**Public Member Functions**

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**)  
throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.608.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3019).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.608.2 Constructor & Destructor Documentation

6.608.2.1 **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::PartialCommandMarshaller**  
( ) [inline]

6.608.2.2 **virtual activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::~~PartialCommandMarshaller**  
( ) [inline, virtual]

### 6.608.3 Member Function Documentation

6.608.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2379).

6.608.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2379).

**6.608.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2380).

**6.608.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).



6.608

**activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller**

**Class Reference**

3025

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2380).

6.608.3.5 **virtual int activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::tightMarshal1**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** )  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2381).

6.608.3.6 **virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::tightMarshal2**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
\* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*,  
**utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** )  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2381).

```
6.608.3.7 virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2382).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h`

## 6.609 **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3023).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/PartialCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller**:

#### Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()

6.609

activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller

Class Reference

3027

- 
- virtual **commands::DataStructure \* createObject ()** const

*Creates a new instance of this marshalable type.*

- virtual unsigned char **getDataStructureType ()** const

*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)**  
throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn)** throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut)** throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.609.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3023).  
NOTE!: This file is auto generated - do not modify! if you need to make a change,  
please see the Java Classes in the activemq-openwire-generator module

6.609.2 Constructor & Destructor Documentation

6.609.2.1 `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::PartialCommandMarshaller ( ) [inline]`

6.609.2.2 `virtual activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::~~PartialCommandMarshaller ( ) [inline, virtual]`

6.609.3 Member Function Documentation

6.609.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller` (p. 2392).

6.609.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller` (p. 2392).

6.609.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides the data sink

6.609

**activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller**

**Class Reference**

**3029**

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2393).

**6.609.3.4** virtual void **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2393).

**6.609.3.5** virtual int **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::tightMarshal1** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2393).

```
6.609.3.6  virtual void activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2394).

```
6.609.3.7  virtual void activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

6.610

**activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller**

**Class Reference**

**3031**

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2394).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**PartialCommandMarshaller.h**

## 6.610 **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3028).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller**:

### Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.610.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3028).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.610.2 Constructor & Destructor Documentation

6.610.2.1 **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::PartialCommandMarshaller**  
( ) [inline]

6.610.2.2 **virtual activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::~~PartialCommandMarshaller**  
( ) [inline, virtual]

### 6.610.3 Member Function Documentation

6.610.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::createObject** ( ) **const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 2396).

6.610.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.



6.610

**activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller**

**Class Reference**

**3033**

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 2397).

**6.610.3.3** virtual void **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::looseMarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 2397).

**6.610.3.4** virtual void **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 2397).

```
6.610.3.5  virtual int activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1690).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 2398).

```
6.610.3.6  virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

### 6.611 decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference 3035

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1698).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 2398).

```
6.610.3.7 virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
  * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1706).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 2399).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h`

### 6.611 decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference

Decaf's implementation of a Smart **Pointer** (p. 3032) that is a template on a Type and is **Thread** (p. 3876) Safe if the default Reference Counter is used.

```
#include <src/main/decaf/lang/Pointer.h>
```

#### Public Types

- `typedef T * PointerType`
- `typedef T & ReferenceType`
- `typedef REFCOUNTER CounterType`

## Public Member Functions

- **Pointer** ()  
*Default Constructor.*
- **Pointer** (const **PointerType** value)  
*Explicit Constructor, creates a **Pointer** (p. 3032) that contains value with a single reference.*
- **Pointer** (const **Pointer** &value) throw ()  
*Copy constructor.*
- template<typename T1 , typename R1 >  
**Pointer** (const **Pointer**< T1, R1 > &value) throw ()  
*Copy constructor.*
- template<typename T1 , typename R1 >  
**Pointer** (const **Pointer**< T1, R1 > &value, const **STATIC\_CAST\_TOKEN** &)  
throw ()  
*Static Cast constructor.*
- template<typename T1 , typename R1 >  
**Pointer** (const **Pointer**< T1, R1 > &value, const **DYNAMIC\_CAST\_TOKEN** &)  
throw ( decaf::lang::exceptions::ClassCastException )  
*Dynamic Cast constructor.*
- virtual ~**Pointer** () throw ()
- void **reset** (T \*value)  
*Resets the **Pointer** (p. 3032) to hold the new value.*
- T \* **release** ()  
*Releases the **Pointer** (p. 3032) held and resets the internal pointer value to Null.*
- **PointerType** **get** () const  
*Gets the real pointer that is contained within this **Pointer** (p. 3032).*
- void **swap** (**Pointer** &value) throw ()  
***Exception** (p. 1886) Safe Swap Function.*
- **Pointer** & **operator=** (const **Pointer** &right) throw ()  
*Assigns the value of right to this **Pointer** (p. 3032) and increments the reference Count.*
- template<typename T1 , typename R1 >  
**Pointer** & **operator=** (const **Pointer**< T1, R1 > &right) throw ()
- **ReferenceType** **operator\*** ()  
*Dereference Operator, returns a reference to the Contained value.*

## 6.611 decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference 3037

- **ReferenceType operator\*** () const
- **PointerType operator->** ()

*Indirection Operator; returns a pointer to the Contained value.*

- **PointerType operator->** () const
- **bool operator!** () const
- `template<typename T1 , typename R1 >`  
**bool operator==** (const **Pointer**< T1, R1 > &right) const
- `template<typename T1 , typename R1 >`  
**bool operator!=** (const **Pointer**< T1, R1 > &right) const
- `template<typename T1 >`  
**Pointer**< T1, **CounterType** > **dynamicCast** () const
- `template<typename T1 >`  
**Pointer**< T1, **CounterType** > **staticCast** () const

### Friends

- **bool operator==** (const **Pointer** &left, const T \*right)
- **bool operator==** (const T \*left, const **Pointer** &right)
- **bool operator!=** (const **Pointer** &left, const T \*right)
- **bool operator!=** (const T \*left, const **Pointer** &right)

### 6.611.1 Detailed Description

`template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter>`  
`class decaf::lang::Pointer< T, REFCOUNTER >`

Decaf's implementation of a Smart **Pointer** (p. 3032) that is a template on a Type and is **Thread** (p. 3876) Safe if the default Reference Counter is used. This **Pointer** (p. 3032) type allows for the substitution of different Reference Counter implementations which provide a means of using invasive reference counting if desired using a custom implementation of `ReferenceCounter`.

The Decaf smart pointer provide comparison operators for comparing **Pointer** (p. 3032) instances in the same manner as normal pointer, except that it does not provide an overload of operators ( <, <=, >, >= ). To allow use of a **Pointer** (p. 3032) in a STL container that requires it, **Pointer** (p. 3032) provides an implementation of `std::less`.

### Since

1.0

## 6.611.2 Member Typedef Documentation

6.611.2.1 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> typedef REFCOUNTER  
decaf::lang::Pointer< T, REFCOUNTER >::CounterType`

6.611.2.2 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> typedef T*  
decaf::lang::Pointer< T, REFCOUNTER >::PointerType`

6.611.2.3 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> typedef T&  
decaf::lang::Pointer< T, REFCOUNTER >::ReferenceType`

## 6.611.3 Constructor & Destructor Documentation

6.611.3.1 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> decaf::lang::Pointer< T,  
REFCOUNTER >::Pointer ( ) [inline]`

Default Constructor.

Initialized the contained pointer to NULL, using the -> operator results in an exception unless reset to contain a real value.

Referenced by `decaf::lang::Pointer< TransactionId >::reset()`.

6.611.3.2 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> decaf::lang::Pointer<  
T, REFCOUNTER >::Pointer ( const PointerType value ) [inline,  
explicit]`

Explicit Constructor, creates a **Pointer** (p. 3032) that contains value with a single reference.

This object now has ownership until a call to release.

### Parameters

<i>value</i>	- instance of the type we are containing here.
--------------	--

6.611.3.3 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> decaf::lang::Pointer< T,  
REFCOUNTER >::Pointer ( const Pointer< T, REFCOUNTER > & value ) throw ()  
[inline]`

Copy constructor.

Copies the value contained in the pointer to the new instance and increments the reference counter.

## 6.611 decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference 3039

6.611.3.4 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 ,  
typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer ( const  
Pointer< T1, R1 > & value ) throw () [inline]`

Copy constructor.

Copies the value contained in the pointer to the new instance and increments the reference counter.

6.611.3.5 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 ,  
typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer ( const  
Pointer< T1, R1 > & value, const STATIC_CAST_TOKEN & ) throw ()  
[inline]`

Static Cast constructor.

Copies the value contained in the pointer to the new instance and increments the reference counter performing a static cast on the value contained in the source **Pointer** (p. 3032) object.

### Parameters

<i>value</i>	- <b>Pointer</b> (p. 3032) instance to cast to this type.
--------------	---

6.611.3.6 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 ,  
typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer ( const  
Pointer< T1, R1 > & value, const DYNAMIC_CAST_TOKEN & ) throw (  
decaf::lang::exceptions::ClassCastException) [inline]`

Dynamic Cast constructor.

Copies the value contained in the pointer to the new instance and increments the reference counter performing a dynamic cast on the value contained in the source **Pointer** (p. 3032) object. If the cast fails and return NULL then this method throws a **ClassCastException**.

### Parameters

<i>value</i>	- <b>Pointer</b> (p. 3032) instance to cast to this type.
--------------	---

### Exceptions

<i>ClassCastException</i>	if the dynamic cast returns NULL
---------------------------	----------------------------------

```
6.611.3.7 template<typename T, typename REFCOUNTER =
    decaf::util::concurrent::atomic::AtomicRefCounter> virtual decaf::lang::Pointer<
    T, REFCOUNTER>::~~Pointer( ) throw() [inline, virtual]
```

#### 6.611.4 Member Function Documentation

```
6.611.4.1 template<typename T, typename REFCOUNTER =
    decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 >
    Pointer<T1, CounterType> decaf::lang::Pointer< T, REFCOUNTER
    >::dynamicCast( ) const [inline]
```

```
6.611.4.2 template<typename T, typename REFCOUNTER =
    decaf::util::concurrent::atomic::AtomicRefCounter> PointerType
    decaf::lang::Pointer< T, REFCOUNTER>::get( ) const [inline]
```

Gets the real pointer that is contained within this **Pointer** (p. 3032).

This is not really safe since the caller could delete or alter the pointer but it mimics the STL `auto_ptr` and gives access in cases where the caller absolutely needs the real **Pointer** (p. 3032). Use at your own risk.

#### Returns

the contained pointer.

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::equals()`, `activemq::state::ConnectionState::getTransactionState()`, `decaf::lang::operator!=()`, `decaf::lang::Pointer< TransactionId >::operator!=()`, `std::less< decaf::lang::Pointer< T >>::operator()`, `decaf::lang::operator==()`, `decaf::lang::Pointer< TransactionId >::operator==()`, and `activemq::state::ConnectionState::removeTempDestination()`.

```
6.611.4.3 template<typename T, typename REFCOUNTER =
    decaf::util::concurrent::atomic::AtomicRefCounter> bool decaf::lang::Pointer<
    T, REFCOUNTER>::operator!( ) const [inline]
```

```
6.611.4.4 template<typename T, typename REFCOUNTER =
    decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 ,
    typename R1 > bool decaf::lang::Pointer< T, REFCOUNTER >::operator!=(
    const Pointer< T1, R1 > & right ) const [inline]
```

```
6.611.4.5 template<typename T, typename REFCOUNTER =
    decaf::util::concurrent::atomic::AtomicRefCounter> ReferenceType
    decaf::lang::Pointer< T, REFCOUNTER>::operator*( ) const [inline]
```

```
6.611.4.6 template<typename T, typename REFCOUNTER =
    decaf::util::concurrent::atomic::AtomicRefCounter> ReferenceType
    decaf::lang::Pointer< T, REFCOUNTER>::operator*( ) [inline]
```

Dereference Operator, returns a reference to the Contained value.



## 6.611 decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference 3041

This method throws an `NullPointerException` if the contained value is `NULL`.

### Returns

reference to the contained pointer.

### Exceptions

<i>NullPointerException</i>	if the contained value is Null
-----------------------------	--------------------------------

```
6.611.4.7  template<typename T, typename REFCOUNTER =  
           decaf::util::concurrent::atomic::AtomicRefCounter> PointerType  
           decaf::lang::Pointer< T, REFCOUNTER >::operator-> ( ) [inline]
```

Indirection Operator, returns a pointer to the Contained value.

This method throws an `NullPointerException` if the contained value is `NULL`.

### Returns

reference to the contained pointer.

### Exceptions

<i>NullPointerException</i>	if the contained value is Null
-----------------------------	--------------------------------

```
6.611.4.8  template<typename T, typename REFCOUNTER =  
           decaf::util::concurrent::atomic::AtomicRefCounter> PointerType  
           decaf::lang::Pointer< T, REFCOUNTER >::operator-> ( ) const [inline]
```

```
6.611.4.9  template<typename T, typename REFCOUNTER =  
           decaf::util::concurrent::atomic::AtomicRefCounter> Pointer&  
           decaf::lang::Pointer< T, REFCOUNTER >::operator= ( const Pointer< T,  
           REFCOUNTER > & right ) throw () [inline]
```

Assigns the value of `right` to this **Pointer** (p. 3032) and increments the reference Count.

### Parameters

<i>right</i>	- <b>Pointer</b> (p. 3032) on the right hand side of an operator= call to this.
--------------	---

```
6.611.4.10  template<typename T, typename REFCOUNTER =
             decaf::util::concurrent::atomic::AtomicRefCount> template<typename T1 ,
             typename R1 > Pointer& decaf::lang::Pointer< T, REFCOUNTER >::operator=
             ( const Pointer< T1, R1 > & right ) throw ()  [inline]
```

```
6.611.4.11  template<typename T, typename REFCOUNTER =
             decaf::util::concurrent::atomic::AtomicRefCount> template<typename T1 ,
             typename R1 > bool decaf::lang::Pointer< T, REFCOUNTER >::operator== (
             const Pointer< T1, R1 > & right ) const  [inline]
```

```
6.611.4.12  template<typename T, typename REFCOUNTER =
             decaf::util::concurrent::atomic::AtomicRefCount> T* decaf::lang::Pointer< T,
             REFCOUNTER >::release ( )  [inline]
```

Releases the **Pointer** (p. 3032) held and resets the internal pointer value to Null.

This method is not guaranteed to be safe if the **Pointer** (p. 3032) is held by more than one object or this method is called from more than one thread.

#### Parameters

<i>value</i>	- The new value to contain.
--------------	-----------------------------

#### Returns

The pointer instance that was held by this **Pointer** (p. 3032) object, the pointer is no longer owned by this **Pointer** (p. 3032) and won't be freed when this **Pointer** (p. 3032) goes out of scope.

Referenced by decaf::lang::Pointer< TransactionId >::Pointer().

```
6.611.4.13  template<typename T, typename REFCOUNTER =
             decaf::util::concurrent::atomic::AtomicRefCount> void decaf::lang::Pointer<
             T, REFCOUNTER >::reset ( T * value )  [inline]
```

Resets the **Pointer** (p. 3032) to hold the new value.

Before the new value is stored reset checks if the old value should be destroyed and if so calls delete. Call reset with a value of NULL is supported and acts to set this **Pointer** (p. 3032) to a NULL pointer.

#### Parameters

<i>value</i>	- The new value to contain.
--------------	-----------------------------

## 6.612 decaf::lang::PointerComparator< T, R > Class Template Reference 3043

- 6.611.4.14 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 >  
Pointer<T1, CounterType> decaf::lang::Pointer< T, REFCOUNTER  
>::staticCast ( ) const [inline]`
- 6.611.4.15 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> void decaf::lang::Pointer<  
T, REFCOUNTER >::swap ( Pointer< T, REFCOUNTER > & value ) throw ()  
[inline]`

**Exception** (p. 1886) Safe Swap Function.

### Parameters

<i>value</i>	- the value to swap with this.
--------------	--------------------------------

Referenced by `decaf::lang::Pointer< TransactionId >::operator=()`, and `decaf::lang::Pointer< TransactionId >::swap()`.

### 6.611.5 Friends And Related Function Documentation

- 6.611.5.1 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> bool operator!= ( const  
Pointer< T, REFCOUNTER > & left, const T * right ) [friend]`
- 6.611.5.2 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> bool operator!= ( const T * left,  
const Pointer< T, REFCOUNTER > & right ) [friend]`
- 6.611.5.3 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> bool operator== ( const  
Pointer< T, REFCOUNTER > & left, const T * right ) [friend]`
- 6.611.5.4 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> bool operator== ( const T * left,  
const Pointer< T, REFCOUNTER > & right ) [friend]`

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Pointer.h`

## 6.612 decaf::lang::PointerComparator< T, R > Class Template Reference

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 3032) instance.

```
#include <src/main/decaf/lang/Pointer.h>
```

Inheritance diagram for decaf::lang::PointerComparator< T, R >:

## Public Member Functions

- virtual bool **operator()** (const **Pointer**< T, R > &left, const **Pointer**< T, R > &right) const
- virtual int **compare** (const **Pointer**< T, R > &left, const **Pointer**< T, R > &right) const

### 6.612.1 Detailed Description

```
template<typename T, typename R = decaf::util::concurrent::atomic::AtomicRefCounter> class
decaf::lang::PointerComparator< T, R >
```

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 3032) instance. This can be useful in the case where a series of values in a Collection is more efficiently accessed in the Objects Natural Order and not the underlying pointers location in memory.

Also this allows **Pointer** (p. 3032) objects that Point to different instances of the same type to be compared based on the comparison of the object itself and not just the value of the pointer.

### 6.612.2 Member Function Documentation

```
6.612.2.1 template<typename T , typename R =
decaf::util::concurrent::atomic::AtomicRefCounter> virtual int
decaf::lang::PointerComparator< T, R >::compare ( const Pointer< T, R >
& left, const Pointer< T, R > & right ) const [inline, virtual]
```

```
6.612.2.2 template<typename T , typename R =
decaf::util::concurrent::atomic::AtomicRefCounter> virtual bool
decaf::lang::PointerComparator< T, R >::operator() ( const Pointer< T, R
> & left, const Pointer< T, R > & right ) const [inline, virtual]
```

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Pointer.h**

## 6.613 activemq::cmsutil::PooledSession Class Reference

A pooled session object that wraps around a delegate session.

```
#include <src/main/activemq/cmsutil/PooledSession.h>
```

Inheritance diagram for activemq::cmsutil::PooledSession:

## Public Member Functions

- **PooledSession** (**SessionPool** \*pool, **cms::Session** \*session)
- virtual **~PooledSession** ()  
*Does nothing.*
- virtual **cms::Session** \* **getSession** ()  
*Returns a non-constant reference to the internal session object.*
- virtual const **cms::Session** \* **getSession** () const  
*Returns a constant reference to the internal session object.*
- virtual void **close** () throw ( cms::CMSEException )  
*Returns this session back to the pool, but does not close or destroy the internal session object.*
- virtual void **commit** () throw ( cms::CMSEException )  
*Commits all messages done in this transaction and releases any locks currently held.*
- virtual void **rollback** () throw ( cms::CMSEException )  
*Rolls back all messages done in this transaction and releases any locks currently held.*
- virtual void **recover** () throw ( cms::CMSEException )  
*Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.*
- virtual **cms::MessageConsumer** \* **createConsumer** (const **cms::Destination** \*destination) throw ( cms::CMSEException )  
*Creates a MessageConsumer for the specified destination.*
- virtual **cms::MessageConsumer** \* **createConsumer** (const **cms::Destination** \*destination, const std::string &selector) throw ( cms::CMSEException )  
*Creates a MessageConsumer for the specified destination, using a message selector.*
- virtual **cms::MessageConsumer** \* **createConsumer** (const **cms::Destination** \*destination, const std::string &selector, bool noLocal) throw ( cms::CMSEException )  
*Creates a MessageConsumer for the specified destination, using a message selector.*
- virtual **cms::MessageConsumer** \* **createDurableConsumer** (const **cms::Topic** \*destination, const std::string &name, const std::string &selector, bool noLocal=false) throw ( cms::CMSEException )

*Creates a durable subscriber to the specified topic, using a Message selector.*

- virtual **cms::MessageConsumer \* createCachedConsumer** (const **cms::Destination** \*destination, const std::string &selector, bool noLocal) throw ( cms::CMSEException )

*First checks the internal consumer cache and creates one if none exist for the given destination, selector, noLocal.*

- virtual **cms::MessageProducer \* createProducer** (const **cms::Destination** \*destination) throw ( cms::CMSEException )

*Creates a MessageProducer to send messages to the specified destination.*

- virtual **cms::MessageProducer \* createCachedProducer** (const **cms::Destination** \*destination) throw ( cms::CMSEException )

*First checks the internal producer cache and creates one if none exist for the given destination.*

- virtual **cms::QueueBrowser \* createBrowser** (const **cms::Queue** \*queue) throw ( cms::CMSEException )

*Creates a new QueueBrowser to peek at Messages on the given Queue.*

- virtual **cms::QueueBrowser \* createBrowser** (const **cms::Queue** \*queue, const std::string &selector) throw ( cms::CMSEException )

*Creates a new QueueBrowser to peek at Messages on the given Queue.*

- virtual **cms::Queue \* createQueue** (const std::string &queueName) throw ( cms::CMSEException )

*Creates a queue identity given a Queue name.*

- virtual **cms::Topic \* createTopic** (const std::string &topicName) throw ( cms::CMSEException )

*Creates a topic identity given a Queue name.*

- virtual **cms::TemporaryQueue \* createTemporaryQueue** () throw ( cms::CMSEException )

*Creates a TemporaryQueue object.*

- virtual **cms::TemporaryTopic \* createTemporaryTopic** () throw ( cms::CMSEException )

*Creates a TemporaryTopic object.*

- virtual **cms::Message \* createMessage** () throw ( cms::CMSEException )

*Creates a new Message.*

- virtual **cms::BytesMessage \* createBytesMessage** () throw ( cms::CMSEException )

*Creates a BytesMessage.*

- virtual **cms::BytesMessage \* createBytesMessage** (const unsigned char \*bytes, int bytesSize) throw ( cms::CMSEException)  
*Creates a BytesMessage and sets the payload to the passed value.*
- virtual **cms::StreamMessage \* createStreamMessage** () throw ( cms::CMSEException )  
*Creates a new StreamMessage.*
- virtual **cms::TextMessage \* createTextMessage** () throw ( cms::CMSEException )  
*Creates a new TextMessage.*
- virtual **cms::TextMessage \* createTextMessage** (const std::string &text) throw ( cms::CMSEException )  
*Creates a new TextMessage and set the text to the value given.*
- virtual **cms::MapMessage \* createMapMessage** () throw ( cms::CMSEException )  
*Creates a new MapMessage.*
- virtual **cms::Session::AcknowledgeMode getAcknowledgeMode** () const throw ( cms::CMSEException )  
*Returns the acknowledgment mode of the session.*
- virtual bool **isTransacted** () const throw ( cms::CMSEException )  
*Gets if the Sessions is a Transacted Session.*
- virtual void **unsubscribe** (const std::string &name) throw ( cms::CMSEException )  
*Unsubscribes a durable subscription that has been created by a client.*

### Protected Member Functions

- **PooledSession** (const **PooledSession** &)
- **PooledSession & operator=** (const **PooledSession** &)

#### 6.613.1 Detailed Description

A pooled session object that wraps around a delegate session. Calls to close this session only result in giving the session back to the pool.

### 6.613.2 Constructor & Destructor Documentation

6.613.2.1 `activemq::cmsutil::PooledSession::PooledSession ( const PooledSession & )`  
[inline, protected]

6.613.2.2 `activemq::cmsutil::PooledSession::PooledSession ( SessionPool * pool, cms::Session * session )`

6.613.2.3 `virtual activemq::cmsutil::PooledSession::~~PooledSession ( )` [virtual]

Does nothing.

### 6.613.3 Member Function Documentation

6.613.3.1 `virtual void activemq::cmsutil::PooledSession::close ( ) throw ( cms::CMSEException )` [virtual]

Returns this session back to the pool, but does not close or destroy the internal session object.

Implements `cms::Session` (p. 3464).

6.613.3.2 `virtual void activemq::cmsutil::PooledSession::commit ( ) throw ( cms::CMSEException )` [inline, virtual]

Commits all messages done in this transaction and releases any locks currently held.

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
<i>IllegalStateException</i>	- if the method is not called by a transacted session.

Implements `cms::Session` (p. 3465).

References `cms::Session::commit()`.

6.613.3.3 `virtual cms::QueueBrowser* activemq::cmsutil::PooledSession::createBrowser ( const cms::Queue * queue, const std::string & selector ) throw ( cms::CMSEException )` [virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

#### Parameters

<i>queue</i>	the Queue to browse
<i>selector</i>	the Message selector to filter which messages are browsed.



**Returns**

New QueueBrowser that is owned by the caller.

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
<i>InvalidDestinationException</i>	- if the destination given is invalid.

Implements **cms::Session** (p. 3466).

**6.613.3.4** virtual **cms::QueueBrowser\*** **activemq::cmsutil::PooledSession::createBrowser** (  
const **cms::Queue** \* *queue* ) throw ( **cms::CMSEException** ) [virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

**Parameters**

<i>queue</i>	the Queue to browse
--------------	---------------------

**Returns**

New QueueBrowser that is owned by the caller.

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
<i>InvalidDestinationException</i>	- if the destination given is invalid.

Implements **cms::Session** (p. 3465).

**6.613.3.5** virtual **cms::BytesMessage\*** **activemq::cmsutil::PooledSession::createBytesMessage** ( ) throw  
( **cms::CMSEException** ) [inline, virtual]

Creates a BytesMessage.

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3466).

**6.613.3.6** `virtual cms::BytesMessage* activemq::cmsutil::PooledSession::createBytesMessage ( const unsigned char * bytes, int bytesSize ) throw ( cms::CMSEException )`  
`[inline, virtual]`

Creates a BytesMessage and sets the payload to the passed value.

#### Parameters

<i>bytes</i>	an array of bytes to set in the message
<i>bytesSize</i>	the size of the bytes array, or number of bytes to use

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3466).

**6.613.3.7** `virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createCachedConsumer ( const cms::Destination * destination, const std::string & selector, bool noLocal ) throw ( cms::CMSEException )`  
`[virtual]`

First checks the internal consumer cache and creates one if none exist for the given destination, selector, noLocal.

If created, the consumer is added to the pool's lifecycle manager.

#### Parameters

<i>destination</i>	the destination to receive on
<i>selector</i>	the selector to use
<i>noLocal</i>	whether or not to receive messages from the same connection

#### Returns

the consumer resource

#### Exceptions

<i>cms::CMSEException</i> (p. 1190)	if something goes wrong.
--	--------------------------

**6.613.3.8** `virtual cms::MessageProducer* activemq::cmsutil::PooledSession::createCachedProducer ( const cms::Destination * destination ) throw ( cms::CMSEException )`  
`[virtual]`

First checks the internal producer cache and creates one if none exist for the given destination.

If created, the producer is added to the pool's lifecycle manager.

#### Parameters

<i>destination</i>	the destination to send on
--------------------	----------------------------

#### Returns

the producer resource

#### Exceptions

<b><i>cms::CMSEException</i></b> (p. 1190)	if something goes wrong.
---	--------------------------

**6.613.3.9** virtual **cms::MessageConsumer\*** **activemq::cmsutil::PooledSession::createConsumer** ( const **cms::Destination** \* *destination* ) throw ( **cms::CMSEException** ) [inline, virtual]

Creates a MessageConsumer for the specified destination.

#### Parameters

<i>destination</i>	the Destination that this consumer receiving messages for.
--------------------	--

#### Returns

pointer to a new MessageConsumer that is owned by the caller ( caller deletes )

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
<i>InvalidDestinationException</i>	- if an invalid destination is specified.

Implements **cms::Session** (p. 3467).

**6.613.3.10** virtual **cms::MessageConsumer\*** **activemq::cmsutil::PooledSession::createConsumer** ( const **cms::Destination** \* *destination*, const std::string & *selector* ) throw ( **cms::CMSEException** ) [inline, virtual]

Creates a MessageConsumer for the specified destination, using a message selector.

#### Parameters

<i>destination</i>	the Destination that this consumer receiving messages for.
<i>selector</i>	the Message Selector to use

**Returns**

pointer to a new MessageConsumer that is owned by the caller ( caller deletes )

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
<i>InvalidDestinationException</i>	- if an invalid destination is specified.
<i>InvalidSelectorException</i>	- if the message selector is invalid.

Implements **cms::Session** (p. 3467).

```
6.613.3.11 virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createConsumer ( const
cms::Destination * destination, const std::string & selector, bool noLocal )
throw ( cms::CMSEException ) [inline, virtual]
```

Creates a MessageConsumer for the specified destination, using a message selector.

**Parameters**

<i>destination</i>	the Destination that this consumer receiving messages for.
<i>selector</i>	the Message Selector to use
<i>noLocal</i>	if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

**Returns**

pointer to a new MessageConsumer that is owned by the caller ( caller deletes )

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
<i>InvalidDestinationException</i>	- if an invalid destination is specified.
<i>InvalidSelectorException</i>	- if the message selector is invalid.

Implements **cms::Session** (p. 3468).

```
6.613.3.12 virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createDurableConsumer (
const cms::Topic * destination, const std::string & name, const std::string &
selector, bool noLocal = false ) throw ( cms::CMSEException ) [inline,
virtual]
```

Creates a durable subscriber to the specified topic, using a Message selector.

Sessions that create durable consumers must use the same client Id as was used the last time the subscription was created in order to receive all messages that were delivered while the client was offline.

#### Parameters

<i>destination</i>	the topic to subscribe to
<i>name</i>	The name used to identify the subscription
<i>selector</i>	the Message Selector to use
<i>noLocal</i>	if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

#### Returns

pointer to a new durable MessageConsumer that is owned by the caller ( caller deletes )

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
<i>InvalidDestinationException</i>	- if an invalid destination is specified.
<i>InvalidSelectorException</i>	- if the message selector is invalid.

Implements **cms::Session** (p. 3469).

**6.613.3.13** virtual **cms::MapMessage\*** **activemq::cmsutil::PooledSession::createMapMessage ( )** throw (**cms::CMSEException**) [inline, virtual]

Creates a new MapMessage.

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3469).

**6.613.3.14** virtual **cms::Message\*** **activemq::cmsutil::PooledSession::createMessage ( )** throw (**cms::CMSEException**) [inline, virtual]

Creates a new Message.

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3470).

**6.613.3.15** `virtual cms::MessageProducer* activemq::cmsutil::PooledSession::createProducer ( const cms::Destination * destination ) throw ( cms::CMSEException )` `[inline, virtual]`

Creates a MessageProducer to send messages to the specified destination.

#### Parameters

<i>destination</i>	the Destination to send on
--------------------	----------------------------

#### Returns

New MessageProducer that is owned by the caller.

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
<i>InvalidDestinationException</i>	- if an invalid destination is specified.

Implements **cms::Session** (p. 3470).

**6.613.3.16** `virtual cms::Queue* activemq::cmsutil::PooledSession::createQueue ( const std::string & queueName ) throw ( cms::CMSEException )` `[inline, virtual]`

Creates a queue identity given a Queue name.

#### Parameters

<i>queueName</i>	the name of the new Queue
------------------	---------------------------

#### Returns

new Queue pointer that is owned by the caller.

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3470).

**6.613.3.17** virtual cms::StreamMessage\* activemq::cmsutil::PooledSession::createStreamMessage ( )  
throw ( cms::CMSEException ) [inline, virtual]

Creates a new StreamMessage.

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements cms::Session (p. 3471).

**6.613.3.18** virtual cms::TemporaryQueue\* activemq::cmsutil::PooledSession::createTemporaryQueue ( )  
throw ( cms::CMSEException ) [inline, virtual]

Creates a TemporaryQueue object.

#### Returns

new TemporaryQueue pointer that is owned by the caller.

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements cms::Session (p. 3471).

**6.613.3.19** virtual cms::TemporaryTopic\* activemq::cmsutil::PooledSession::createTemporaryTopic ( )  
throw ( cms::CMSEException ) [inline, virtual]

Creates a TemporaryTopic object.

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements cms::Session (p. 3471).

**6.613.3.20** virtual cms::TextMessage\* activemq::cmsutil::PooledSession::createTextMessage ( const  
std::string & text ) throw ( cms::CMSEException ) [inline, virtual]

Creates a new TextMessage and set the text to the value given.

#### Parameters

<i>text</i>	the initial text for the message
-------------	----------------------------------

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3472).

**6.613.3.21** `virtual cms::TextMessage* activemq::cmsutil::PooledSession::createTextMessage ( ) throw ( cms::CMSEException ) [inline, virtual]`

Creates a new TextMessage.

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3472).

**6.613.3.22** `virtual cms::Topic* activemq::cmsutil::PooledSession::createTopic ( const std::string & topicName ) throw ( cms::CMSEException ) [inline, virtual]`

Creates a topic identity given a Queue name.

**Parameters**

<i>topicName</i>	the name of the new Topic
------------------	---------------------------

**Returns**

new Topic pointer that is owned by the caller.

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3472).

**6.613.3.23** `virtual cms::Session::AcknowledgeMode activemq::cmsutil::PooledSession::getAcknowledgeMode ( ) const throw ( cms::CMSEException ) [inline, virtual]`

Returns the acknowledgment mode of the session.

**Returns**

the Sessions Acknowledge Mode



**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3473).

**6.613.3.24** `virtual cms::Session* activemq::cmsutil::PooledSession::getSession ( )`  
`[inline, virtual]`

Returns a non-constant reference to the internal session object.

**Returns**

the session object.

**6.613.3.25** `virtual const cms::Session* activemq::cmsutil::PooledSession::getSession ( )`  
`const [inline, virtual]`

Returns a constant reference to the internal session object.

**Returns**

the session object.

**6.613.3.26** `virtual bool activemq::cmsutil::PooledSession::isTransacted ( ) const throw (`  
`cms::CMSEException) [inline, virtual]`

Gets if the Sessions is a Transacted Session.

**Returns**

transacted true - false.

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3473).

**6.613.3.27** `PooledSession& activemq::cmsutil::PooledSession::operator= ( const`  
`PooledSession & ) [inline, protected]`

**6.613.3.28** `virtual void activemq::cmsutil::PooledSession::recover ( ) throw (`  
`cms::CMSEException) [inline, virtual]`

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.

All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "re-delivered"
- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

#### Exceptions

<i>CMSEException</i>	- if the CMS provider fails to stop and restart message delivery due to some internal error.
<i>IllegalStateException</i>	- if the method is called by a transacted session.

Implements **cms::Session** (p. 3474).

**6.613.3.29** `virtual void activemq::cmsutil::PooledSession::rollback ( ) throw ( cms::CMSEException ) [inline, virtual]`

Rolls back all messages done in this transaction and releases any locks currently held.

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
<i>IllegalStateException</i>	- if the method is not called by a transacted session.

Implements **cms::Session** (p. 3474).

**6.613.3.30** `virtual void activemq::cmsutil::PooledSession::unsubscribe ( const std::string & name ) throw ( cms::CMSEException ) [inline, virtual]`

Unsubscribes a durable subscription that has been created by a client.

This method deletes the state being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active MessageConsumer or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

#### Parameters

<i>name</i>	The name used to identify this subscription
-------------	---

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3475).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**PooledSession.h**

**6.614 decaf::util::concurrent::PooledThread Class Reference**

```
#include <src/main/decaf/util/concurrent/PooledThread.h>
```

Inheritance diagram for decaf::util::concurrent::PooledThread:

**Public Member Functions**

- **PooledThread** (**ThreadPool** \*pool)

*Constructor.*

- virtual **~PooledThread** ()
- virtual void **run** ()

*Run Method for this object waits for something to be enqueued on the **ThreadPool** (p. 3888) and then grabs it and calls its run method.*

- virtual void **stop** () throw ( lang::Exception )

*Stops the Thread, thread will complete its task if currently running one, and then die.*

- virtual bool **isBusy** ()

*Checks to see if the thread is busy, if busy it means that this thread has taken a task from the ThreadPool's queue and is processing it.*

- virtual void **setPooledThreadListener** (**PooledThreadListener** \*listener)

*Adds a listener to this **PooledThread** (p. 3056) to be notified when this thread starts and completes a task.*

- virtual **PooledThreadListener** \* **getPooledThreadListener** ()

*Removes a listener for this **PooledThread** (p. 3056) to be notified when this thread starts and completes a task.*

### 6.614.1 Constructor & Destructor Documentation

#### 6.614.1.1 `decaf::util::concurrent::PooledThread::PooledThread ( ThreadPool * pool )`

Constructor.

##### Parameters

<i>pool</i>	the parant <b>ThreadPool</b> (p. 3888) object
-------------	---

#### 6.614.1.2 `virtual decaf::util::concurrent::PooledThread::~~PooledThread ( ) [virtual]`

### 6.614.2 Member Function Documentation

#### 6.614.2.1 `virtual PooledThreadListener* decaf::util::concurrent::PooledThread::getPooledThreadListener ( ) [inline, virtual]`

Removes a listener for this **PooledThread** (p. 3056) to be notified when this thread starts and completes a task.

##### Returns

a pointer to this thread's listener or NULL

#### 6.614.2.2 `virtual bool decaf::util::concurrent::PooledThread::isBusy ( ) [inline, virtual]`

Checks to see if the thread is busy, if busy it means that this thread has taken a task from the ThreadPool's queue and is processing it.

##### Returns

true if the Thread is busy

#### 6.614.2.3 `virtual void decaf::util::concurrent::PooledThread::run ( ) [virtual]`

Run Method for this object waits for something to be enqueued on the **ThreadPool** (p. 3888) and then grabs it and calls its run method.

Reimplemented from **decaf::lang::Thread** (p. 3884).

#### 6.614.2.4 `virtual void decaf::util::concurrent::PooledThread::setPooledThreadListener ( PooledThreadListener * listener ) [inline, virtual]`

Adds a listener to this **PooledThread** (p. 3056) to be notified when this thread starts and completes a task.

**Parameters**

<i>listener</i>	the listener to send notifications to.
-----------------	--

**6.614.2.5** virtual void decaf::util::concurrent::PooledThread::stop ( ) throw ( lang::Exception ) [virtual]

Stops the Thread, thread will complete its task if currently running one, and then die.

Does not block.

**Exceptions**

<i>Exception</i>	
------------------	--

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**PooledThread.h**

**6.615 decaf::util::concurrent::PooledThreadListener Class Reference**

Abstract Listener Interface for users of **ThreadPool** (p. 3888).

```
#include <src/main/decaf/util/concurrent/PooledThreadListener.h>
```

Inheritance diagram for decaf::util::concurrent::PooledThreadListener:

**Public Member Functions**

- virtual ~**PooledThreadListener** ()
- virtual void **onTaskStarted** (**PooledThread** \*thread)=0  
*Called by a pooled thread when it is about to begin executing a new task.*
- virtual void **onTaskCompleted** (**PooledThread** \*thread)=0  
*Called by a pooled thread when it has completed a task and is going back to waiting for another task to run.*
- virtual void **onTaskException** (**PooledThread** \*thread, lang::Exception &ex)=0  
*Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 3056) is now no longer running.*

### 6.615.1 Detailed Description

Abstract Listener Interface for users of **ThreadPool** (p. 3888). The implementor of this class receives events related to the execution and termination of threads running in the **ThreadPool** (p. 3888).

Since

1.0

### 6.615.2 Constructor & Destructor Documentation

6.615.2.1 `virtual decaf::util::concurrent::PooledThreadListener::~~PooledThreadListener ( )`  
[inline, virtual]

### 6.615.3 Member Function Documentation

6.615.3.1 `virtual void decaf::util::concurrent::PooledThreadListener::onTaskCompleted ( PooledThread * thread )` [pure virtual]

Called by a pooled thread when it has completed a task and is going back to waiting for another task to run.

Parameters

<i>thread</i>	- Pointer the the Pooled Thread that is making this call.
---------------	---

Implemented in **decaf::util::concurrent::ThreadPool** (p. 3892).

6.615.3.2 `virtual void decaf::util::concurrent::PooledThreadListener::onTaskException ( PooledThread * thread, lang::Exception & ex )` [pure virtual]

Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 3056) is now no longer running.

Parameters

<i>thread</i>	- Pointer to the Pooled Thread that is making this call
<i>ex</i>	- The Exception that occurred.

Implemented in **decaf::util::concurrent::ThreadPool** (p. 3893).

6.615.3.3 `virtual void decaf::util::concurrent::PooledThreadListener::onTaskStarted ( PooledThread * thread )` [pure virtual]

Called by a pooled thread when it is about to begin executing a new task.

**Parameters**

<i>thread</i>	- Pointer to the Pooled Thread that is making this call
---------------	---

Implemented in **decaf::util::concurrent::ThreadPool** (p. 3893).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**PooledThreadListener.h**

**6.616 decaf::net::PortUnreachableException Class Reference**

```
#include <src/main/decaf/net/PortUnreachableException.h>
```

Inheritance diagram for decaf::net::PortUnreachableException:

**Public Member Functions**

- **PortUnreachableException** () throw ()

*Default Constructor.*

- **PortUnreachableException** (const Exception &ex) throw ()

*Conversion Constructor from some other Exception.*

- **PortUnreachableException** (const **PortUnreachableException** &ex) throw ()

*Copy Constructor.*

- **PortUnreachableException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()

*Constructor - Initializes the file name and line number where this message occurred.*

- **PortUnreachableException** (const std::exception \*cause) throw ()

*Constructor.*

- **PortUnreachableException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()

*Constructor - Initializes the file name and line number where this message occurred.*

- virtual **PortUnreachableException** \* **clone** () const

*Clones this exception.*

- virtual ~**PortUnreachableException** () throw ()

### 6.616.1 Constructor & Destructor Documentation

**6.616.1.1** `decaf::net::PortUnreachableException::PortUnreachableException ( ) throw ()`  
[inline]

Default Constructor.

**6.616.1.2** `decaf::net::PortUnreachableException::PortUnreachableException ( const Exception & ex ) throw ()` [inline]

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.616.1.3** `decaf::net::PortUnreachableException::PortUnreachableException ( const PortUnreachableException & ex ) throw ()` [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.616.1.4** `decaf::net::PortUnreachableException::PortUnreachableException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()`  
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.616.1.5** `decaf::net::PortUnreachableException::PortUnreachableException ( const std::exception * cause ) throw ()` [inline]

Constructor.



**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.616.1.6** `decaf::net::PortUnreachableException::PortUnreachableException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.616.1.7** `virtual decaf::net::PortUnreachableException::~~PortUnreachableException ( ) throw () [inline, virtual]`

**6.616.2 Member Function Documentation**

**6.616.2.1** `virtual PortUnreachableException* decaf::net::PortUnreachableException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns**

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 3629).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/PortUnreachableException.h`

**6.617 activemq::core::PrefetchPolicy Class Reference**

Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.

```
#include <src/main/activemq/core/PrefetchPolicy.h>
```

Inheritance diagram for `activemq::core::PrefetchPolicy`:

### Public Member Functions

- virtual `~PrefetchPolicy ()`
- virtual void `setDurableTopicPrefetch (int value)=0`  
*Sets the amount of prefetched messages for a Durable Topic.*
- virtual int `getDurableTopicPrefetch () const =0`  
*Gets the amount of messages to prefetch for a Durable Topic.*
- virtual void `setQueuePrefetch (int value)=0`  
*Sets the amount of prefetched messages for a Queue.*
- virtual int `getQueuePrefetch () const =0`  
*Gets the amount of messages to prefetch for a Queue.*
- virtual void `setQueueBrowserPrefetch (int value)=0`  
*Sets the amount of prefetched messages for a Queue Browser.*
- virtual int `getQueueBrowserPrefetch () const =0`  
*Gets the amount of messages to prefetch for a Queue Browser.*
- virtual void `setTopicPrefetch (int value)=0`  
*Sets the amount of prefetched messages for a Topic.*
- virtual int `getTopicPrefetch () const =0`  
*Gets the amount of messages to prefetch for a Topic.*
- virtual int `getMaxPrefetchLimit (int value) const =0`  
*Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.*
- virtual `PrefetchPolicy * clone () const =0`  
*Clone the Policy and return a new pointer to that clone.*
- virtual void `configure (const decaf::util::Properties &properties)`  
*Checks the supplied properties object for properties matching the configurable settings of this class.*

### Protected Member Functions

- `PrefetchPolicy ()`

### 6.617.1 Detailed Description

Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.

#### Since

3.2.0

### 6.617.2 Constructor & Destructor Documentation

**6.617.2.1** `activemq::core::PrefetchPolicy::PrefetchPolicy ( )` `[protected]`

**6.617.2.2** `virtual activemq::core::PrefetchPolicy::~~PrefetchPolicy ( )` `[virtual]`

### 6.617.3 Member Function Documentation

**6.617.3.1** `virtual PrefetchPolicy* activemq::core::PrefetchPolicy::clone ( ) const` `[pure virtual]`

Clone the Policy and return a new pointer to that clone.

#### Returns

pointer to a new **PrefetchPolicy** (p. 3062) instance that is a clone of this one.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1727).

**6.617.3.2** `virtual void activemq::core::PrefetchPolicy::configure ( const decaf::util::Properties & properties )` `[virtual]`

Checks the supplied properties object for properties matching the configurable settings of this class.

The default implementation looks for properties named with the prefix `cms.PrefetchPolicy.XXX` where XXX is the name of a property with a public setter method. For instance `cms.PrefetchPolicy.topicPrefetch` will be used to set the value of the topic prefetch limit.

Subclasses can override this method to add more configuration options or to exclude certain parameters from being set via the properties object.

#### Parameters

<i>properties</i>	The Properties object used to configure this object.
-------------------	--

#### Exceptions

<i>NumberFormatException</i>	if a property that is numeric cannot be converted
------------------------------	---

<i>IllegalArgumentEx- ception</i>	if a property can't be converted to the correct type.
---------------------------------------	---

**6.617.3.3** `virtual int activemq::core::PrefetchPolicy::getDurableTopicPrefetch ( ) const`  
`[pure virtual]`

Gets the amount of messages to prefetch for a Durable Topic.

#### Returns

value of the number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1728).

**6.617.3.4** `virtual int activemq::core::PrefetchPolicy::getMaxPrefetchLimit ( int value ) const`  
`[pure virtual]`

Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.

#### Returns

the allowable value for a prefetch limit, either requested or the max.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1728).

**6.617.3.5** `virtual int activemq::core::PrefetchPolicy::getQueueBrowserPrefetch ( ) const`  
`[pure virtual]`

Gets the amount of messages to prefetch for a Queue Browser.

#### Returns

value of the number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1728).

**6.617.3.6** `virtual int activemq::core::PrefetchPolicy::getQueuePrefetch ( ) const` `[pure virtual]`

Gets the amount of messages to prefetch for a Queue.

#### Returns

value of the number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1728).

**6.617.3.7** `virtual int activemq::core::PrefetchPolicy::getTopicPrefetch ( ) const` `[pure virtual]`

Gets the amount of messages to prefetch for a Topic.

#### Returns

value of the number of messages to prefetch.

Implemented in `activemq::core::policies::DefaultPrefetchPolicy` (p. 1729).

**6.617.3.8** `virtual void activemq::core::PrefetchPolicy::setDurableTopicPrefetch ( int value )` `[pure virtual]`

Sets the amount of prefetched messages for a Durable Topic.

#### Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implemented in `activemq::core::policies::DefaultPrefetchPolicy` (p. 1729).

**6.617.3.9** `virtual void activemq::core::PrefetchPolicy::setQueueBrowserPrefetch ( int value )` `[pure virtual]`

Sets the amount of prefetched messages for a Queue Browser.

#### Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implemented in `activemq::core::policies::DefaultPrefetchPolicy` (p. 1729).

**6.617.3.10** `virtual void activemq::core::PrefetchPolicy::setQueuePrefetch ( int value )` `[pure virtual]`

Sets the amount of prefetched messages for a Queue.

#### Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implemented in `activemq::core::policies::DefaultPrefetchPolicy` (p. 1729).

**6.617.3.11** `virtual void activemq::core::PrefetchPolicy::setTopicPrefetch ( int value )` [pure virtual]

Sets the amount of prefetched messages for a Topic.

#### Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1730).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/PrefetchPolicy.h`

## 6.618 activemq::util::PrimitiveList Class Reference

List of primitives.

```
#include <src/main/activemq/util/PrimitiveList.h>
```

Inheritance diagram for `activemq::util::PrimitiveList`:

### Public Member Functions

- **PrimitiveList** ()

*Default Constructor; creates an Empty list.*

- virtual **~PrimitiveList** ()

- **PrimitiveList** (const **decaf::util::List**< **PrimitiveValueNode** > &src)

*Copy Constructor.*

- **PrimitiveList** (const **PrimitiveList** &src)

*Copy Constructor.*

- std::string **toString** () const

*Converts the contents into a formatted string that can be output in a Log File or other debugging tool.*

- virtual bool **getBool** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )

*Gets the Boolean value at the specified index.*

- virtual void **setBool** (std::size\_t index, bool value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*

- virtual unsigned char **getByte** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )

*Gets the Byte value at the specified index.*

- virtual void **setByte** (std::size\_t index, unsigned char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*

- virtual char **getChar** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )

*Gets the Character value at the specified index.*

- virtual void **setChar** (std::size\_t index, char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*

- virtual short **getShort** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )

*Gets the Short value at the specified index.*

- virtual void **setShort** (std::size\_t index, short value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*

- virtual int **getInt** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )

*Gets the Integer value at the specified index.*

- virtual void **setInt** (std::size\_t index, int value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*

- virtual long long **getLong** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )

*Gets the Long value at the specified index.*

- virtual void **setLong** (std::size\_t index, long long value) throw ( decaf::lang::exceptions::IndexOutOfBoundsEx  
)  
*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*
- virtual float **getFloat** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsEx  
decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Float value at the specified index.*
- virtual void **setFloat** (std::size\_t index, float value) throw ( decaf::lang::exceptions::IndexOutOfBoundsEx  
)  
*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*
- virtual double **getDouble** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsEx  
decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Double value at the specified index.*
- virtual void **setDouble** (std::size\_t index, double value) throw ( decaf::lang::exceptions::IndexOutOfBoundsEx  
)  
*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*
- virtual std::string **getString** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsEx  
decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the String value at the specified index.*
- virtual void **setString** (std::size\_t index, const std::string &value) throw ( decaf::lang::exceptions::IndexOutOfBoundsEx  
)  
*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*
- virtual std::vector< unsigned char > **getByteArray** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsEx  
decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Byte Array value at the specified index.*
- virtual void **setByteArray** (std::size\_t index, const std::vector< unsigned char > &value) throw ( decaf::lang::exceptions::IndexOutOfBoundsEx  
)  
*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*



### 6.618.1 Detailed Description

List of primitives.

### 6.618.2 Constructor & Destructor Documentation

#### 6.618.2.1 `activemq::util::PrimitiveList::PrimitiveList ( )`

Default Constructor, creates an Empty list.

#### 6.618.2.2 `virtual activemq::util::PrimitiveList::~~PrimitiveList ( ) [virtual]`

#### 6.618.2.3 `activemq::util::PrimitiveList::PrimitiveList ( const decaf::util::List< PrimitiveValueNode > & src )`

Copy Constructor.

##### Parameters

<i>src</i>	- the Decaf List of PrimitiveNodeValues to copy
------------	---

#### 6.618.2.4 `activemq::util::PrimitiveList::PrimitiveList ( const PrimitiveList & src )`

Copy Constructor.

##### Parameters

<i>src</i>	- the <b>PrimitiveList</b> (p. 3067) to copy
------------	--

### 6.618.3 Member Function Documentation

#### 6.618.3.1 `virtual bool activemq::util::PrimitiveList::getBool ( std::size_t index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Gets the Boolean value at the specified index.

##### Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

##### Returns

value contained at the given index

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index is > <b>size()</b> (p. 3708)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

**6.618.3.2** `virtual unsigned char activemq::util::PrimitiveList::getBytes ( std::size_t index )  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Gets the Byte value at the specified index.

**Parameters**

<i>index</i>	- index to get value from
--------------	---------------------------

**Returns**

value contained at the given index

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index is > <b>size()</b> (p. 3708)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

**6.618.3.3** `virtual std::vector<unsigned char> activemq::util::PrimitiveList::getBytesArray  
( std::size_t index ) const throw ( de-  
cafe::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Gets the Byte Array value at the specified index.

**Parameters**

<i>index</i>	- index to get value from
--------------	---------------------------

**Returns**

value contained at the given index

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index is > <b>size()</b> (p. 3708)
----------------------------------	---------------------------------------

<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.
--------------------------------------	--

**6.618.3.4** `virtual char activemq::util::PrimitiveList::getChar ( std::size_t index ) const  
throw ( decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Gets the Character value at the specified index.

#### Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

#### Returns

value contained at the given index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is > <b>size()</b> (p. 3708)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

**6.618.3.5** `virtual double activemq::util::PrimitiveList::getDouble ( std::size_t index )  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Gets the Double value at the specified index.

#### Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

#### Returns

value contained at the given index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is > <b>size()</b> (p. 3708)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

**6.618.3.6** `virtual float activemq::util::PrimitiveList::getFloat ( std::size_t index ) const`  
`throw ( decaf::lang::exceptions::IndexOutOfBoundsException,`  
`decaf::lang::exceptions::UnsupportedOperationException )`  
`[virtual]`

Gets the Float value at the specified index.

#### Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

#### Returns

value contained at the given index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is > <b>size()</b> (p. 3708)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

**6.618.3.7** `virtual int activemq::util::PrimitiveList::getInt ( std::size_t index ) const`  
`throw ( decaf::lang::exceptions::IndexOutOfBoundsException,`  
`decaf::lang::exceptions::UnsupportedOperationException )`  
`[virtual]`

Gets the Integer value at the specified index.

#### Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

#### Returns

value contained at the given index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is > <b>size()</b> (p. 3708)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

**6.618.3.8** `virtual long long activemq::util::PrimitiveList::getLong ( std::size_t index )  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Gets the Long value at the specified index.

#### Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

#### Returns

value contained at the given index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is > <b>size()</b> (p. 3708)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

**6.618.3.9** `virtual short activemq::util::PrimitiveList::getShort ( std::size_t index ) const  
throw ( decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Gets the Short value at the specified index.

#### Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

#### Returns

value contained at the given index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is > <b>size()</b> (p. 3708)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

**6.618.3.10** `virtual std::string activemq::util::PrimitiveList::getString ( std::size_t index )  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Gets the String value at the specified index.

#### Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

#### Returns

value contained at the given index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is > <b>size()</b> (p. 3708)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

**6.618.3.11** `virtual void activemq::util::PrimitiveList::setBool ( std::size_t index, bool value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

#### Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index > <b>size()</b> (p. 3708).
----------------------------------	-------------------------------------

**6.618.3.12** `virtual void activemq::util::PrimitiveList::setByte ( std::size_t index, unsigned char value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

**Parameters**

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index > <b>size()</b> (p. 3708).
----------------------------------	-------------------------------------

**6.618.3.13** `virtual void activemq::util::PrimitiveList::setByteArray ( std::size_t index, const std::vector< unsigned char > & value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

**Parameters**

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index > <b>size()</b> (p. 3708).
----------------------------------	-------------------------------------

**6.618.3.14** `virtual void activemq::util::PrimitiveList::setChar ( std::size_t index, char value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

**Parameters**

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index > <b>size()</b> (p. 3708).
----------------------------------	-------------------------------------

**6.618.3.15** `virtual void activemq::util::PrimitiveList::setDouble ( std::size_t index, double value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

#### Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index > <b>size()</b> (p. 3708).
----------------------------------	-------------------------------------

**6.618.3.16** `virtual void activemq::util::PrimitiveList::setFloat ( std::size_t index, float value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

#### Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index > <b>size()</b> (p. 3708).
----------------------------------	-------------------------------------

**6.618.3.17** `virtual void activemq::util::PrimitiveList::setInt ( std::size_t index, int value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

#### Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index



**Exceptions**

<i>IndexOutOfBoundsException</i>	if index > <b>size()</b> (p. 3708).
----------------------------------	-------------------------------------

**6.618.3.18** virtual void activemq::util::PrimitiveList::setLong ( std::size\_t *index*, long long *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

**Parameters**

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index > <b>size()</b> (p. 3708).
----------------------------------	-------------------------------------

**6.618.3.19** virtual void activemq::util::PrimitiveList::setShort ( std::size\_t *index*, short *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

**Parameters**

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index > <b>size()</b> (p. 3708).
----------------------------------	-------------------------------------

**6.618.3.20** virtual void activemq::util::PrimitiveList::setString ( std::size\_t *index*, const std::string & *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the

index is greater than the size of the list.

#### Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index > <b>size()</b> (p. 3708).
----------------------------------	-------------------------------------

#### 6.618.3.21 `std::string activemq::util::PrimitiveList::toString ( ) const`

Converts the contents into a formatted string that can be output in a Log File or other debugging tool.

#### Returns

formatted String of all elements in the list.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/PrimitiveList.h`

### 6.619 `activemq::util::PrimitiveMap` Class Reference

Map of named primitives.

```
#include <src/main/activemq/util/PrimitiveMap.h>
```

Inheritance diagram for `activemq::util::PrimitiveMap`:

#### Public Member Functions

- **PrimitiveMap** ()  
*Default Constructor; creates an empty map.*
- virtual **~PrimitiveMap** ()
- **PrimitiveMap** (const **decaf::util::Map**< std::string, **PrimitiveValueNode** > &source)  
*Copy Constructor.*
- **PrimitiveMap** (const **PrimitiveMap** &source)  
*Copy Constructor.*

- std::string **toString** () const

*Converts the contents into a formatted string that can be output in a Log File or other debugging tool.*

- virtual bool **getBool** (const std::string &key) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )

*Gets the Boolean value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*

- virtual void **setBool** (const std::string &key, bool value)

*Sets the value at key to the specified type.*

- virtual unsigned char **getByte** (const std::string &key) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )

*Gets the Byte value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*

- virtual void **setByte** (const std::string &key, unsigned char value)

*Sets the value at key to the specified type.*

- virtual char **getChar** (const std::string &key) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )

*Gets the Character value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*

- virtual void **setChar** (const std::string &key, char value)

*Sets the value at key to the specified type.*

- virtual short **getShort** (const std::string &key) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )

*Gets the Short value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*

- virtual void **setShort** (const std::string &key, short value)

*Sets the value at key to the specified type.*

- virtual int **getInt** (const std::string &key) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )

*Gets the Integer value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*

- virtual void **setInt** (const std::string &key, int value)

*Sets the value at key to the specified type.*

- virtual long long **getLong** (const std::string &key) const throw ( decaf::lang::exceptions::NoSuchElementException  
decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Long value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*
- virtual void **setLong** (const std::string &key, long long value)  
*Sets the value at key to the specified type.*
- virtual float **getFloat** (const std::string &key) const throw ( decaf::lang::exceptions::NoSuchElementException  
decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Float value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*
- virtual void **setFloat** (const std::string &key, float value)  
*Sets the value at key to the specified type.*
- virtual double **getDouble** (const std::string &key) const throw ( decaf::lang::exceptions::NoSuchElementException  
decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Double value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*
- virtual void **setDouble** (const std::string &key, double value)  
*Sets the value at key to the specified type.*
- virtual std::string **getString** (const std::string &key) const throw ( decaf::lang::exceptions::NoSuchElement  
decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the String value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*
- virtual void **setString** (const std::string &key, const std::string &value)  
*Sets the value at key to the specified type.*
- virtual std::vector< unsigned char > **getByteArray** (const std::string &key)  
const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOp  
)  
*Gets the Byte Array value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*
- virtual void **setByteArray** (const std::string &key, const std::vector< unsigned  
char > &value)  
*Sets the value at key to the specified type.*

### 6.619.1 Detailed Description

Map of named primitives.

## 6.619.2 Constructor & Destructor Documentation

### 6.619.2.1 activemq::util::PrimitiveMap::PrimitiveMap ( )

Default Constructor, creates an empty map.

### 6.619.2.2 virtual activemq::util::PrimitiveMap::~~PrimitiveMap ( ) [virtual]

### 6.619.2.3 activemq::util::PrimitiveMap::PrimitiveMap ( const decaf::util::Map< std::string, PrimitiveValueNode > & source )

Copy Constructor.

#### Parameters

<i>source</i>	The Decaf Library Map instance whose elements will be copied into this Map.
---------------	---

### 6.619.2.4 activemq::util::PrimitiveMap::PrimitiveMap ( const PrimitiveMap & source )

Copy Constructor.

#### Parameters

<i>source</i>	The <b>PrimitiveMap</b> (p. 3079) whose elements will be copied into this Map.
---------------	--

## 6.619.3 Member Function Documentation

### 6.619.3.1 virtual bool activemq::util::PrimitiveMap::getBool ( const std::string & key ) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]

Gets the Boolean value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

#### Parameters

<i>key</i>	- the location to return the value from.
------------	--

#### Returns

the value at key in the type requested.

#### Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
-------------------------------	---------------------------

<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns
--------------------------------------	--

**6.619.3.2** `virtual unsigned char activemq::util::PrimitiveMap::getByte ( const std::string & key ) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )`  
 [virtual]

Gets the Byte value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

#### Parameters

<i>key</i>	- the location to return the value from.
------------	--

#### Returns

the value at key in the type requested.

#### Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

**6.619.3.3** `virtual std::vector<unsigned char> activemq::util::PrimitiveMap::getByteArray ( const std::string & key ) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )`  
 [virtual]

Gets the Byte Array value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

#### Parameters

<i>key</i>	- the location to return the value from.
------------	--

#### Returns

the value at key in the type requested.

#### Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
-------------------------------	---------------------------

<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns
--------------------------------------	--

**6.619.3.4** `virtual char activemq::util::PrimitiveMap::getChar ( const std::string & key )  
const throw ( decaf::lang::exceptions::NoSuchElementException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Gets the Character value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

#### Parameters

<i>key</i>	- the location to return the value from.
------------	--

#### Returns

the value at key in the type requested.

#### Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

**6.619.3.5** `virtual double activemq::util::PrimitiveMap::getDouble ( const std::string & key  
) const throw ( decaf::lang::exceptions::NoSuchElementException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Gets the Double value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

#### Parameters

<i>key</i>	- the location to return the value from.
------------	--

#### Returns

the value at key in the type requested.

#### Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

**6.619.3.6** `virtual float activemq::util::PrimitiveMap::getFloat ( const std::string & key )  
const throw ( decaf::lang::exceptions::NoSuchElementException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Gets the Float value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

#### Parameters

<i>key</i>	- the location to return the value from.
------------	--

#### Returns

the value at key in the type requested.

#### Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

**6.619.3.7** `virtual int activemq::util::PrimitiveMap::getInt ( const std::string & key )  
const throw ( decaf::lang::exceptions::NoSuchElementException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Gets the Integer value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

#### Parameters

<i>key</i>	- the location to return the value from.
------------	--

#### Returns

the value at key in the type requested.

#### Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns



**6.619.3.8** virtual long long activemq::util::PrimitiveMap::getLong ( const std::string & key ) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]

Gets the Long value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

#### Parameters

<i>key</i>	- the location to return the value from.
------------	--

#### Returns

the value at key in the type requested.

#### Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

**6.619.3.9** virtual short activemq::util::PrimitiveMap::getShort ( const std::string & key ) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]

Gets the Short value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

#### Parameters

<i>key</i>	- the location to return the value from.
------------	--

#### Returns

the value at key in the type requested.

#### Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

**6.619.3.10** `virtual std::string activemq::util::PrimitiveMap::getString ( const std::string & key )  
const throw ( decaf::lang::exceptions::NoSuchElementException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Gets the String value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

#### Parameters

<i>key</i>	- the location to return the value from.
------------	--

#### Returns

the value at key in the type requested.

#### Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnSupportedOperationException</i>	if the value cannot be converted to the type this method returns

**6.619.3.11** `virtual void activemq::util::PrimitiveMap::setBool ( const std::string & key, bool  
value ) [virtual]`

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

#### Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

**6.619.3.12** `virtual void activemq::util::PrimitiveMap::setByte ( const std::string & key, unsigned  
char value ) [virtual]`

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

#### Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

**6.619.3.13** `virtual void activemq::util::PrimitiveMap::setByteArray ( const std::string & key,  
const std::vector< unsigned char > & value ) [virtual]`

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

#### Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

**6.619.3.14** `virtual void activemq::util::PrimitiveMap::setChar ( const std::string & key, char  
value ) [virtual]`

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

#### Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

**6.619.3.15** `virtual void activemq::util::PrimitiveMap::setDouble ( const std::string & key, double  
value ) [virtual]`

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

#### Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

**6.619.3.16** `virtual void activemq::util::PrimitiveMap::setFloat ( const std::string & key, float  
value ) [virtual]`

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

#### Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

**6.619.3.17** `virtual void activemq::util::PrimitiveMap::setInt ( const std::string & key, int value )`  
[virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

#### Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

**6.619.3.18** `virtual void activemq::util::PrimitiveMap::setLong ( const std::string & key, long long value )`  
[virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

#### Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

**6.619.3.19** `virtual void activemq::util::PrimitiveMap::setShort ( const std::string & key, short value )`  
[virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

#### Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

**6.619.3.20** `virtual void activemq::util::PrimitiveMap::setString ( const std::string & key, const std::string & value )`  
[virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

#### Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

#### 6.619.3.21 std::string activemq::util::PrimitiveMap::toString ( ) const

Converts the contents into a formatted string that can be output in a Log File or other debugging tool.

#### Returns

formatted String of all elements in the map.

The documentation for this class was generated from the following file:

- src/main/activemq/util/**PrimitiveMap.h**

## 6.620 activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller Class Reference

This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

```
#include <src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h>
```

### Public Member Functions

- **PrimitiveTypesMarshaller** ( )
- virtual ~**PrimitiveTypesMarshaller** ( )

### Static Public Member Functions

- static void **marshal** (const **util::PrimitiveMap** \*map, std::vector< unsigned char > &buffer) throw ( decaf::lang::Exception )  
*Marshal a primitive map object to the given byte buffer.*
- static void **unmarshal** (**util::PrimitiveMap** \*map, const std::vector< unsigned char > &buffer) throw ( decaf::lang::Exception )  
*Unmarshal a PrimitiveMap from the provided Byte buffer.*
- static void **marshal** (const **util::PrimitiveList** \*list, std::vector< unsigned char > &buffer) throw ( decaf::lang::Exception )  
*Marshal a primitive list object to the given byte buffer.*
- static void **unmarshal** (**util::PrimitiveList** \*list, const std::vector< unsigned char > &buffer) throw ( decaf::lang::Exception )  
*Unmarshal a PrimitiveList from the provided byte buffer.*
- static void **marshalMap** (const **util::PrimitiveMap** \*map, **decaf::io::DataOutputStream** &dataOut) throw ( decaf::lang::Exception )

*Marshal a primitive map object to the given DataOutputStream.*

- static **util::PrimitiveMap \* unmarshalMap (decaf::io::DataInputStream &dataIn)**  
throw ( decaf::lang::Exception )

*Unmarshal a PrimitiveMap from the provided DataInputStream.*

- static void **marshalList (const util::PrimitiveList \*list, decaf::io::DataOutputStream &dataOut)** throw ( decaf::lang::Exception )

*Marshal a PrimitiveList to the given DataOutputStream.*

- static **util::PrimitiveList \* unmarshalList (decaf::io::DataInputStream &dataIn)**  
throw ( decaf::lang::Exception )

*Unmarshal a PrimitiveList from the given DataInputStream.*

### Static Protected Member Functions

- static void **marshalPrimitiveMap (decaf::io::DataOutputStream &dataOut, const decaf::util::Map< std::string, util::PrimitiveValueNode > &map)** throw ( decaf::io::IOException )

*Marshal a Map of Primitives to the given OutputStream, can result in recursive calls to this method if the map contains maps of maps.*

- static void **marshalPrimitiveList (decaf::io::DataOutputStream &dataOut, const decaf::util::List< util::PrimitiveValueNode > &list)** throw ( decaf::io::IOException )

*Marshal a List of Primitives to the given OutputStream, can result in recursive calls to this method if the list contains lists of lists.*

- static void **marshalPrimitive (decaf::io::DataOutputStream &dataOut, const util::PrimitiveValueNode &value)** throw ( decaf::io::IOException )

*Used to Marshal the Primitive types out on the Wire.*

- static void **unmarshalPrimitiveMap (decaf::io::DataInputStream &dataIn, util::PrimitiveMap &map)** throw ( decaf::io::IOException )

*Unmarshals a Map of Primitives from the given InputStream, can result in recursive calls to this method if the map contains maps of maps.*

- static void **unmarshalPrimitiveList (decaf::io::DataInputStream &dataIn, decaf::util::StlList< util::PrimitiveValueNode > &list)** throw ( decaf::io::IOException )

*Unmarshals a List of Primitives from the given InputStream, can result in recursive calls to this method if the list contains lists of lists.*

- static **util::PrimitiveValueNode unmarshalPrimitive (decaf::io::DataInputStream &dataIn)** throw ( decaf::io::IOException )

*Unmarshals a Primitive Type from the stream, and returns it as a value Node.*

## 6.620.1 Detailed Description

This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

## 6.620.2 Constructor & Destructor Documentation

- 6.620.2.1    `activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::PrimitiveTypesMarshaller`  
           ( ) [inline]
- 6.620.2.2    `virtual activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::~~PrimitiveTypesMarshaller`  
           ( ) [inline, virtual]

## 6.620.3 Member Function Documentation

- 6.620.3.1    `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshal`  
           ( const util::PrimitiveMap \* *map*, std::vector< unsigned char > & *buffer* ) throw  
           ( decaf::lang::Exception ) [static]

Marshal a primitive map object to the given byte buffer.

### Parameters

<i>map</i>	Map to Marshal.
<i>buffer</i>	The byte buffer to write the marshaled data to.

### Exceptions

<i>Exception</i>	if an error occurs during the marshaling process.
------------------	---

- 6.620.3.2    `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshal`  
           ( const util::PrimitiveList \* *list*, std::vector< unsigned char > & *buffer* ) throw (   
           decaf::lang::Exception ) [static]

Marshal a primitive list object to the given byte buffer.

### Parameters

<i>map</i>	The PrimitiveList to Marshal.
<i>buffer</i>	The byte buffer to write the marshaled data to.

### Exceptions

<i>Exception</i>	if an error occurs during the marshaling process.
------------------	---

**6.620.3.3** `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalList  
( const util::PrimitiveList * list, decaf::io::DataOutputStream & dataOut )  
throw ( decaf::lang::Exception ) [static]`

Marshal a PrimitiveList to the given DataOutputStream.

#### Parameters

<i>list</i>	The list object to Marshal
<i>dataOut</i>	Reference to a DataOutputStream to write the marshaled data to.

#### Exceptions

<i>Exception</i>	if an error occurs during the marshaling process.
------------------	---

**6.620.3.4** `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalMap  
( const util::PrimitiveMap * map, decaf::io::DataOutputStream & dataOut  
) throw ( decaf::lang::Exception ) [static]`

Marshal a primitive map object to the given DataOutputStream.

#### Parameters

<i>map</i>	Map to Marshal.
<i>dataOut</i>	Reference to a DataOutputStream to write the marshaled data to.

#### Exceptions

<i>Exception</i>	if an error occurs during the marshaling process.
------------------	---

**6.620.3.5** `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitive  
( decaf::io::DataOutputStream & dataOut, const util::PrimitiveValueNode  
& value ) throw ( decaf::io::IOException ) [static, protected]`

Used to Marshal the Primitive types out on the Wire.

#### Parameters

<i>dataOut</i>	- the DataOutputStream to write to
<i>value</i>	- the ValueNode to write.

#### Exceptions

<i>IOException</i>	
--------------------	--



**6.620.3.6** static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitiveList  
 ( decaf::io::DataOutputStream & *dataOut*, const decaf::util::List<  
 util::PrimitiveValueNode > & *list* ) throw ( decaf::io::IOException )  
 [static, protected]

Marshal a List of Primitives to the given OutputStream, can result in recursive calls to this method if the list contains lists of lists.

#### Parameters

<i>dataOut</i>	- the DataOutputStream to write to
<i>list</i>	- the ValueNode to write.

#### Exceptions

<i>IOException</i>	
--------------------	--

**6.620.3.7** static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitiveMap  
 ( decaf::io::DataOutputStream & *dataOut*, const decaf::util::Map<  
 std::string, util::PrimitiveValueNode > & *map* ) throw ( decaf::io::IOException ) [static, protected]

Marshal a Map of Primitives to the given OutputStream, can result in recursive calls to this method if the map contains maps of maps.

#### Parameters

<i>dataOut</i>	- the DataOutputStream to write to
<i>map</i>	- the ValueNode to write.

#### Exceptions

<i>IOException</i>	
--------------------	--

**6.620.3.8** static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshal  
 ( util::PrimitiveList \* *list*, const std::vector< unsigned char > & *buffer* ) throw ( decaf::lang::Exception ) [static]

Unmarshal a PrimitiveList from the provided byte buffer.

#### Parameters

<i>map</i>	The List to populate with values from the marshaled data.
<i>buffer</i>	The byte buffer containing the marshaled Map.

#### Exceptions

<i>Exception</i>	if an error occurs during the unmarshal process.
------------------	--

**6.620.3.9** `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshal  
( util::PrimitiveMap * map, const std::vector< unsigned char > & buffer ) throw  
( decaf::lang::Exception ) [static]`

Unmarshal a PrimitiveMap from the provided Byte buffer.

#### Parameters

<i>map</i>	The Map to populate with values from the marshaled data.
<i>buffer</i>	The byte buffer containing the marshaled Map.

#### Exceptions

<i>Exception</i>	if an error occurs during the unmarshal process.
------------------	--

**6.620.3.10** `static util::PrimitiveList* activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalList  
( decaf::io::DataInputStream & dataIn ) throw ( decaf::lang::Exception )  
[static]`

Unmarshal a PrimitiveList from the given DataInputStream.

#### Parameters

<i>dataIn</i>	The DataInputStream instance to read the marshaled PrimitiveList from.
---------------	--

#### Returns

a pointer to a newly allocated PrimitiveList instnace.

#### Exceptions

<i>Exception</i>	if an error occurs during the unmarshal process.
------------------	--

**6.620.3.11** `static util::PrimitiveMap* ac-  
tivismq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalMap ( decaf::io::DataInputStream & dataIn ) throw ( decaf::lang::Exception )  
[static]`

Unmarshal a PrimitiveMap from the provided DataInputStream.

#### Parameters

<i>dataIn</i>	The DataInputStream instance to read the marshaled PrimitiveMap from.
---------------	---

#### Returns

a pointer to a newly allocated PrimitiveMap instnace.

## Exceptions

<i>Exception</i>	if an error occurs during the unmarshal process.
------------------	--

**6.620.3.12** static util::PrimitiveValueNode activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitive ( decaf::io::DataInputStream & *dataIn* ) throw ( decaf::io::IOException )  
 [static, protected]

Unmarshals a Primitive Type from the stream, and returns it as a value Node.

## Parameters

<i>dataIn</i>	- DataInputStream to read from.
---------------	---------------------------------

## Returns

a PrimitiveValueNode containing the data.

## Exceptions

<i>IOException</i>	
--------------------	--

**6.620.3.13** static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitiveList ( decaf::io::DataInputStream & *dataIn*, decaf::util::StlList< util::PrimitiveValueNode > & *list* ) throw ( decaf::io::IOException )  
 [static, protected]

Unmarshals a List of Primitives from the given InputStream, can result in recursive calls to this method if the list contains lists of lists.

## Parameters

<i>dataIn</i>	- DataInputStream to read from.
<i>list</i>	- the ValueNode to write.

## Exceptions

<i>IOException</i>	
--------------------	--

**6.620.3.14** static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitiveMap ( decaf::io::DataInputStream & *dataIn*, util::PrimitiveMap & *map* ) throw ( decaf::io::IOException ) [static, protected]

Unmarshals a Map of Primitives from the given InputStream, can result in recursive calls to this method if the map contains maps of maps.

**Parameters**

<i>dataIn</i>	- DataInputStream to read from.
<i>map</i>	- the map to fill with data.

**Exceptions**

<i>IOException</i>
--------------------

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/**PrimitiveTypesMarshaller.h**

## 6.621 activemq::util::PrimitiveValueNode::PrimitiveValue Union Reference

Define a union type comprised of the various types.

```
#include <src/main/activemq/util/PrimitiveValueNode.h>
```

**Data Fields**

- bool **boolValue**
- unsigned char **byteValue**
- char **charValue**
- short **shortValue**
- int **intValue**
- long long **longValue**
- double **doubleValue**
- float **floatValue**
- std::string \* **stringValue**
- std::vector< unsigned char > \* **byteArrayValue**
- decaf::util::List< **PrimitiveValueNode** > \* **listValue**
- decaf::util::Map< std::string, **PrimitiveValueNode** > \* **mapValue**

### 6.621.1 Detailed Description

Define a union type comprised of the various types.

### 6.621.2 Field Documentation

- 6.621.2.1 `bool activemq::util::PrimitiveValueNode::PrimitiveValue::boolValue`
- 6.621.2.2 `std::vector<unsigned char>* activemq::util::PrimitiveValueNode::PrimitiveValue::byteArrayValue`
- 6.621.2.3 `unsigned char activemq::util::PrimitiveValueNode::PrimitiveValue::byteValue`
- 6.621.2.4 `char activemq::util::PrimitiveValueNode::PrimitiveValue::charValue`
- 6.621.2.5 `double activemq::util::PrimitiveValueNode::PrimitiveValue::doubleValue`
- 6.621.2.6 `float activemq::util::PrimitiveValueNode::PrimitiveValue::floatValue`
- 6.621.2.7 `int activemq::util::PrimitiveValueNode::PrimitiveValue::intValue`
- 6.621.2.8 `decaf::util::List<PrimitiveValueNode>* activemq::util::PrimitiveValueNode::PrimitiveValue::listValue`
- 6.621.2.9 `long long activemq::util::PrimitiveValueNode::PrimitiveValue::longValue`
- 6.621.2.10 `decaf::util::Map<std::string, PrimitiveValueNode>* activemq::util::PrimitiveValueNode::PrimitiveValue::mapValue`
- 6.621.2.11 `short activemq::util::PrimitiveValueNode::PrimitiveValue::shortValue`
- 6.621.2.12 `std::string* activemq::util::PrimitiveValueNode::PrimitiveValue::stringValue`

The documentation for this union was generated from the following file:

- `src/main/activemq/util/PrimitiveValueNode.h`

## 6.622 activemq::util::PrimitiveValueConverter Class Reference

Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 3099) from one type to another.

```
#include <src/main/activemq/util/PrimitiveValueConverter.h>
```

### Public Member Functions

- **PrimitiveValueConverter** ()
- virtual **~PrimitiveValueConverter** ()
- template<typename TO >  
TO **convert** (const **PrimitiveValueNode** &value) const throw ( decaf::lang::exceptions::UnsupportedOperationException )

### 6.622.1 Detailed Description

Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 3099) from one type to another. If the conversion is supported then calling the convert method will throw an `UnsupportedOperationException` to indicate that its not possible to perform the conversion.

This class is used to implement the rules of conversion on CMS Message properties, the following conversion table must be implemented. A value written as the row type can be read in the column type.

	boolean	byte	short	int	long	float	double	String	
boolean	X	X							
byte	X	X	X	X	X				
short	X	X	X	X	X				
int	X	X	X	X	X				
long	X								
float	X	X	X			X	X		
double	X	X				X	X		
String						X	X	X	X

Since

3.0

### 6.622.2 Constructor & Destructor Documentation

6.622.2.1 `activemq::util::PrimitiveValueConverter::PrimitiveValueConverter ( )` [inline]

6.622.2.2 `virtual activemq::util::PrimitiveValueConverter::~~PrimitiveValueConverter ( )`  
[inline, virtual]

### 6.622.3 Member Function Documentation

6.622.3.1 `std::vector< unsigned char > activemq::util::PrimitiveValueConverter::convert ( const PrimitiveValueNode & value ) const throw ( decaf::lang::exceptions::UnsupportedOperationException )`  
[inline]

The documentation for this class was generated from the following file:

- `src/main/activemq/util/PrimitiveValueConverter.h`

## 6.623 activemq::util::PrimitiveValueNode Class Reference

Class that wraps around a single value of one of the many types.

```
#include <src/main/activemq/util/PrimitiveValueNode.h>
```

### Data Structures

- union **PrimitiveValue**

*Define a union type comprised of the various types.*

## Public Types

- enum **PrimitiveType** {  
    **NULL\_TYPE** = 0, **BOOLEAN\_TYPE** = 1, **BYTE\_TYPE** = 2, **CHAR\_TYPE** = 3,  
    **SHORT\_TYPE** = 4, **INTEGER\_TYPE** = 5, **LONG\_TYPE** = 6, **DOUBLE\_TYPE** = 7,  
    **FLOAT\_TYPE** = 8, **STRING\_TYPE** = 9, **BYTE\_ARRAY\_TYPE** = 10, **MAP\_TYPE** = 11,  
    **LIST\_TYPE** = 12, **BIG\_STRING\_TYPE** = 13 }  
    *Enumeration for the various primitive types.*

## Public Member Functions

- **PrimitiveValueNode** ()  
    *Default Constructor, creates a value of the NULL\_TYPE.*
- **PrimitiveValueNode** (bool value)  
    *Boolean Value Constructor.*
- **PrimitiveValueNode** (unsigned char value)  
    *Byte Value Constructor.*
- **PrimitiveValueNode** (char value)  
    *Char Value Constructor.*
- **PrimitiveValueNode** (short value)  
    *Short Value Constructor.*
- **PrimitiveValueNode** (int value)  
    *Int Value Constructor.*
- **PrimitiveValueNode** (long long value)  
    *Long Value Constructor.*
- **PrimitiveValueNode** (float value)  
    *Float Value Constructor.*
- **PrimitiveValueNode** (double value)  
    *Double Value Constructor.*
- **PrimitiveValueNode** (const char \*value)  
    *String Value Constructor.*

- **PrimitiveValueNode** (const std::string &value)  
*String Value Constructor.*
- **PrimitiveValueNode** (const std::vector< unsigned char > &value)  
*Byte Array Value Constructor.*
- **PrimitiveValueNode** (const decaf::util::List< **PrimitiveValueNode** > &value)  
*Primitive List Constructor.*
- **PrimitiveValueNode** (const decaf::util::Map< std::string, **PrimitiveValueNode** > &value)  
*Primitive Map Value Constructor.*
- **PrimitiveValueNode** (const **PrimitiveValueNode** &node)  
*Copy constructor.*
- **~PrimitiveValueNode** ()
- **PrimitiveValueNode & operator=** (const **PrimitiveValueNode** &node)  
*Assignment operator, copies the data from the other node.*
- **bool operator==** (const **PrimitiveValueNode** &node) const  
*Comparison Operator, compares this node to the other node.*
- **PrimitiveType getType** () const  
*Gets the Value Type of this type wrapper.*
- **PrimitiveValue getValue** () const  
*Gets the internal Primitive Value object from this wrapper.*
- **void setValue** (const **PrimitiveValue** &value, **PrimitiveType** valueType)  
*Sets the internal PrimitiveVale object to the new value along with the tag for the type that it consists of.*
- **void clear** ()  
*Clears the value from this wrapper converting it back to a blank NULL\_TYPE value.*
- **void setBool** (bool value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- **bool getBool** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets the Boolean value of this Node.*
- **void setByte** (unsigned char value)



*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*

- unsigned char **getByte** () const throw ( decaf::lang::exceptions::NoSuchElementException )

*Gets the Byte value of this Node.*

- void **setChar** (char value)

*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*

- char **getChar** () const throw ( decaf::lang::exceptions::NoSuchElementException )

*Gets the Character value of this Node.*

- void **setShort** (short value)

*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*

- short **getShort** () const throw ( decaf::lang::exceptions::NoSuchElementException )

*Gets the Short value of this Node.*

- void **setInt** (int value)

*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*

- int **getInt** () const throw ( decaf::lang::exceptions::NoSuchElementException )

*Gets the Integer value of this Node.*

- void **setLong** (long long value)

*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*

- long long **getLong** () const throw ( decaf::lang::exceptions::NoSuchElementException )

*Gets the Long value of this Node.*

- void **setFloat** (float value)

*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*

- float **getFloat** () const throw ( decaf::lang::exceptions::NoSuchElementException )

*Gets the Float value of this Node.*

- void **setDouble** (double value)

*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*

- double **getDouble** () const throw (decaf::lang::exceptions::NoSuchElementException )

*Gets the Double value of this Node.*

- void **setString** (const std::string &value)

*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*

- std::string **getString** () const throw ( decaf::lang::exceptions::NoSuchElementException )

*Gets the String value of this Node.*

- void **setByteArray** (const std::vector< unsigned char > &value)

*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*

- std::vector< unsigned char > **getByteArray** () const throw ( decaf::lang::exceptions::NoSuchElementException )

*Gets the Byte Array value of this Node.*

- void **setList** (const decaf::util::List< PrimitiveValueNode > &value)

*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*

- const decaf::util::List< PrimitiveValueNode > & **getList** () const throw ( decaf::lang::exceptions::NoSuchElementException )

*Gets the Primitive List value of this Node.*

- void **setMap** (const decaf::util::Map< std::string, PrimitiveValueNode > &value)

*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*

- const decaf::util::Map< std::string, PrimitiveValueNode > & **getMap** () const throw ( decaf::lang::exceptions::NoSuchElementException )

*Gets the Primitive Map value of this Node.*

- std::string **toString** () const

*Creates a string representation of this value.*

### 6.623.1 Detailed Description

Class that wraps around a single value of one of the many types. Manages memory for complex types, such as strings. Note: the destructor was left non-virtual so no virtual

table will be created. This probably isn't necessary, but will avoid needless memory allocation. Since we'll never extend this class, not having a virtual destructor isn't a concern.

### 6.623.2 Member Enumeration Documentation

#### 6.623.2.1 enum activemq::util::PrimitiveValueNode::PrimitiveType

Enumeration for the various primitive types.

##### Enumerator:

*NULL\_TYPE*  
*BOOLEAN\_TYPE*  
*BYTE\_TYPE*  
*CHAR\_TYPE*  
*SHORT\_TYPE*  
*INTEGER\_TYPE*  
*LONG\_TYPE*  
*DOUBLE\_TYPE*  
*FLOAT\_TYPE*  
*STRING\_TYPE*  
*BYTE\_ARRAY\_TYPE*  
*MAP\_TYPE*  
*LIST\_TYPE*  
*BIG\_STRING\_TYPE*

### 6.623.3 Constructor & Destructor Documentation

#### 6.623.3.1 activemq::util::PrimitiveValueNode::PrimitiveValueNode ( )

Default Constructor, creates a value of the *NULL\_TYPE*.

#### 6.623.3.2 activemq::util::PrimitiveValueNode::PrimitiveValueNode ( bool *value* )

Boolean Value Constructor.

##### Parameters

<i>value</i>	- the new value to store.
--------------	---------------------------

**6.623.3.3    `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( unsigned char value )`**

Byte Value Constructor.

**Parameters**

<i>value</i>	- the new value to store.
--------------	---------------------------

**6.623.3.4    `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( char value )`**

Char Value Constructor.

**Parameters**

<i>value</i>	- the new value to store.
--------------	---------------------------

**6.623.3.5    `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( short value )`**

Short Value Constructor.

**Parameters**

<i>value</i>	- the new value to store.
--------------	---------------------------

**6.623.3.6    `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( int value )`**

Int Value Constructor.

**Parameters**

<i>value</i>	- the new value to store.
--------------	---------------------------

**6.623.3.7    `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( long long value )`**

Long Value Constructor.

**Parameters**

<i>value</i>	- the new value to store.
--------------	---------------------------

**6.623.3.8    `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( float value )`**

Float Value Constructor.

**Parameters**

<i>value</i>	- the new value to store.
--------------	---------------------------

**6.623.3.9 activemq::util::PrimitiveValueNode::PrimitiveValueNode ( double *value* )**

Double Value Constructor.

**Parameters**

<i>value</i>	- the new value to store.
--------------	---------------------------

**6.623.3.10 activemq::util::PrimitiveValueNode::PrimitiveValueNode ( const char \* *value* )**

String Value Constructor.

**Parameters**

<i>value</i>	- the new value to store.
--------------	---------------------------

**6.623.3.11 activemq::util::PrimitiveValueNode::PrimitiveValueNode ( const std::string & *value* )**

String Value Constructor.

**Parameters**

<i>value</i>	- the new value to store.
--------------	---------------------------

**6.623.3.12 activemq::util::PrimitiveValueNode::PrimitiveValueNode ( const std::vector< unsigned char > & *value* )**

Byte Array Value Constructor.

**Parameters**

<i>value</i>	- the new value to store.
--------------	---------------------------

**6.623.3.13 activemq::util::PrimitiveValueNode::PrimitiveValueNode ( const decaf::util::List< PrimitiveValueNode > & *value* )**

Primitive List Constructor.

**Parameters**

<i>value</i>	- the new value to store.
--------------	---------------------------

**6.623.3.14** `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( const decaf::util::Map< std::string, PrimitiveValueNode > & value )`

Primitive Map Value Constructor.

#### Parameters

<i>value</i>	- the new value to store.
--------------	---------------------------

**6.623.3.15** `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( const PrimitiveValueNode & node )`

Copy constructor.

#### Parameters

<i>node</i>	The instance of another node to copy to this one.
-------------	---

**6.623.3.16** `activemq::util::PrimitiveValueNode::~~PrimitiveValueNode ( ) [inline]`

### 6.623.4 Member Function Documentation

**6.623.4.1** `void activemq::util::PrimitiveValueNode::clear ( )`

Clears the value from this wrapper converting it back to a blank NULL\_TYPE value.

**6.623.4.2** `bool activemq::util::PrimitiveValueNode::getBool ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Boolean value of this Node.

#### Returns

value contained at the given index

#### Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

**6.623.4.3** `unsigned char activemq::util::PrimitiveValueNode::getByte ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Byte value of this Node.

**Returns**

value contained at the given index

**Exceptions**

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

**6.623.4.4** `std::vector<unsigned char> activemq::util::PrimitiveValueNode::getByteArray ( )  
const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Byte Array value of this Node.

**Returns**

value contained at the given index

**Exceptions**

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

**6.623.4.5** `char activemq::util::PrimitiveValueNode::getChar ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Character value of this Node.

**Returns**

value contained at the given index

**Exceptions**

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

**6.623.4.6** `double activemq::util::PrimitiveValueNode::getDouble ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Double value of this Node.

**Returns**

value contained at the given index

**Exceptions**

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

#### 6.623.4.7 `float activemq::util::PrimitiveValueNode::getFloat ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Float value of this Node.

##### Returns

value contained at the given index

##### Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

#### 6.623.4.8 `int activemq::util::PrimitiveValueNode::getInt ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Integer value of this Node.

##### Returns

value contained at the given index

##### Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

#### 6.623.4.9 `const decaf::util::List<PrimitiveValueNode>& activemq::util::PrimitiveValueNode::getList ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Primitive List value of this Node.

##### Returns

value contained at the given index

##### Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---



**6.623.4.10** `long long activemq::util::PrimitiveValueNode::getLong ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Long value of this Node.

#### Returns

value contained at the given index

#### Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

**6.623.4.11** `const decaf::util::Map<std::string, PrimitiveValueNode>& activemq::util::PrimitiveValueNode::getMap ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Primitive Map value of this Node.

#### Returns

value contained at the given index

#### Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

**6.623.4.12** `short activemq::util::PrimitiveValueNode::getShort ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Short value of this Node.

#### Returns

value contained at the given index

#### Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

**6.623.4.13** `std::string activemq::util::PrimitiveValueNode::getString ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the String value of this Node.

**Returns**

value contained at the given index

**Exceptions**

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

**6.623.4.14 PrimitiveType** activemq::util::PrimitiveValueNode::getType ( ) const  
[inline]

Gets the Value Type of this type wrapper.

**Returns**

the PrimitiveType value for this wrapper.

**6.623.4.15 PrimitiveValue** activemq::util::PrimitiveValueNode::getValue ( ) const  
[inline]

Gets the internal Primitive Value object from this wrapper.

**Returns**

a copy of the contained **PrimitiveValue** (p. 3097)

**6.623.4.16 PrimitiveValueNode&** activemq::util::PrimitiveValueNode::operator= ( const  
**PrimitiveValueNode & node** )

Assignment operator, copies the data from the other node.

**Parameters**

<i>node</i>	The instance of another node to copy to this one.
-------------	---

**6.623.4.17 bool** activemq::util::PrimitiveValueNode::operator== ( const **PrimitiveValueNode**  
& *node* ) const

Comparison Operator, compares this node to the other node.

**Returns**

true if the values are the same false otherwise.

**6.623.4.18 void activemq::util::PrimitiveValueNode::setBool ( bool *value* )**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters**

<i>value</i>	- the new value to assign to the element at index
--------------	---

**6.623.4.19 void activemq::util::PrimitiveValueNode::setByte ( unsigned char *value* )**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters**

<i>value</i>	- the new value to assign to the element at index
--------------	---

**6.623.4.20 void activemq::util::PrimitiveValueNode::setByteArray ( const std::vector< unsigned char > & *value* )**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters**

<i>value</i>	- the new value to assign to the element at index
--------------	---

**6.623.4.21 void activemq::util::PrimitiveValueNode::setChar ( char *value* )**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters**

<i>value</i>	- the new value to assign to the element at index
--------------	---

**6.623.4.22 void activemq::util::PrimitiveValueNode::setDouble ( double *value* )**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters**

<i>value</i>	- the new value to assign to the element at index
--------------	---

**6.623.4.23 void activemq::util::PrimitiveValueNode::setFloat ( float *value* )**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters**

<i>value</i>	- the new value to assign to the element at index
--------------	---

**6.623.4.24 void activemq::util::PrimitiveValueNode::setInt ( int *value* )**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters**

<i>value</i>	- the new value to assign to the element at index
--------------	---

**6.623.4.25 void activemq::util::PrimitiveValueNode::setList ( const decaf::util::List< PrimitiveValueNode > & *value* )**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters**

<i>value</i>	- the new value to assign to the element at index
--------------	---

**6.623.4.26 void activemq::util::PrimitiveValueNode::setLong ( long long *value* )**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters**

<i>value</i>	- the new value to assign to the element at index
--------------	---

**6.623.4.27 void activemq::util::PrimitiveValueNode::setMap ( const decaf::util::Map< std::string, PrimitiveValueNode > & *value* )**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters**

<i>value</i>	- the new value to assign to the element at index
--------------	---

**6.623.4.28 void activemq::util::PrimitiveValueNode::setShort ( short *value* )**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters**

<i>value</i>	- the new value to assign to the element at index
--------------	---

**6.623.4.29 void activemq::util::PrimitiveValueNode::setString ( const std::string & *value* )**

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters**

<i>value</i>	- the new value to assign to the element at index
--------------	---

**6.623.4.30 void activemq::util::PrimitiveValueNode::setValue ( const PrimitiveValue & *value*, PrimitiveType *valueType* )**

Sets the internal PrimitiveVale object to the new value along with the tag for the type that it consists of.

**Parameters**

<i>value</i>	The value to set as the value contained in this Node.
<i>valueType</i>	The type of the value being set into this one.

**6.623.4.31 std::string activemq::util::PrimitiveValueNode::toString ( ) const**

Creates a string representation of this value.

**Returns**

string value of this type wrapper.

The documentation for this class was generated from the following file:

- src/main/activemq/util/**PrimitiveValueNode.h**

**6.624 decaf::security::Principal Class Reference**

Base interface for a principal, which can represent an individual or organization.

```
#include <src/main/decaf/security/Principal.h>
```

Inheritance diagram for `decaf::security::Principal`:

### Public Member Functions

- virtual `~Principal()`
- virtual bool **`equals`** (const **`Principal`** &another) const =0  
*Compares two principals to see if they are the same.*
- virtual std::string **`getName`** () const =0  
*Provides the name of this principal.*

#### 6.624.1 Detailed Description

Base interface for a principal, which can represent an individual or organization.

#### 6.624.2 Constructor & Destructor Documentation

6.624.2.1 virtual `decaf::security::Principal::~Principal()` [`inline`, `virtual`]

#### 6.624.3 Member Function Documentation

6.624.3.1 virtual bool `decaf::security::Principal::equals ( const Principal & another )` const  
[`pure virtual`]

Compares two principals to see if they are the same.

##### Parameters

<i>another</i>	A principal to be tested for equality to this one.
----------------	--

##### Returns

true if the given principal is equivalent to this one.

6.624.3.2 virtual std::string `decaf::security::Principal::getName ()` const [`pure virtual`]

Provides the name of this principal.

##### Returns

the name of this principal.

Implemented in `decaf::security::auth::x500::X500Principal` (p. 4140).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**Principal.h**

## 6.625 decaf::util::PriorityQueue< E > Class Template Reference

An unbounded priority queue based on a binary heap algorithm.

```
#include <src/main/decaf/util/PriorityQueue.h>
```

Inheritance diagram for decaf::util::PriorityQueue< E >:

### Data Structures

- class **ConstPriorityQueueIterator**
- class **PriorityQueueIterator**

### Public Member Functions

- **PriorityQueue ()**  
*Creates a Priority **Queue** (p. 3239) with the default initial capacity.*
- **PriorityQueue (std::size\_t initialCapacity)**  
*Creates a Priority **Queue** (p. 3239) with the capacity value supplied.*
- **PriorityQueue (std::size\_t initialCapacity, Comparator< E > \*comparator)**  
*Creates a Priority **Queue** (p. 3239) with the default initial capacity.*
- **PriorityQueue (const Collection< E > &source)**  
*Creates a **PriorityQueue** (p. 3116) containing the elements in the specified **Collection** (p. 1216).*
- **PriorityQueue (const PriorityQueue< E > &source)**  
*Creates a **PriorityQueue** (p. 3116) containing the elements in the specified priority queue.*
- virtual **~PriorityQueue ()**
- **PriorityQueue< E > & operator= (const Collection< E > &source)**  
*Assignment operator; assign another **Collection** (p. 1216) to this one.*
- **PriorityQueue< E > & operator= (const PriorityQueue< E > &source)**  
*Assignment operator; assign another **PriorityQueue** (p. 3116) to this one.*
- virtual **decaf::util::Iterator< E > \* iterator ()**

- virtual **decaf::util::Iterator**< E > \* **iterator** () const
- virtual std::size\_t **size** () const  
*Returns the number of elements in this collection.*
- virtual void **clear** () throw ( lang::exceptions::UnsupportedOperationException )  
*Removes all elements of the queue.*
- virtual bool **offer** (const E &value) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Inserts the specified element into the queue provided that the condition allows such an operation.*
- virtual bool **poll** (E &result)  
*Gets and removes the element in the head of the queue.*
- virtual bool **peek** (E &result) const  
*Gets but not removes the element in the head of the queue.*
- virtual E **remove** () throw ( decaf::lang::exceptions::NoSuchElementException )  
*Retrieves and removes the head of this queue.*
- virtual bool **remove** (const E &value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )  
*Removes a single instance of the specified element from this collection, if it is present (optional operation).*
- virtual bool **add** (const E &value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )  
*Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an IllegalStateException if no space is currently available.*
- **decaf::lang::Pointer**< **Comparator**< E > > **comparator** () const  
*obtains a Copy of the Pointer instance that this **PriorityQueue** (p.3116) is using to compare the elements in the queue with.*

## Friends

- class **PriorityQueueIterator**



### 6.625.1 Detailed Description

`template<typename E> class decaf::util::PriorityQueue< E >`

An unbounded priority queue based on a binary heap algorithm. The elements of the priority queue are ordered according to their natural ordering, or by a **Comparator** (p. 1251) provided to one of the constructors that accepts Comparators. A priority queue relying on natural ordering also does not permit insertion of non-comparable objects (doing so may result in a compiler error).

The head of this queue is the least element with respect to the specified ordering. If multiple elements are tied for least value, the head is one of those elements -- ties are broken arbitrarily. The queue retrieval operations poll, remove, peek, and element access the element at the head of the queue.

A priority queue is unbounded, but has an internal capacity governing the size of an array used to store the elements on the queue. It is always at least as large as the queue size. As elements are added to a priority queue, its capacity grows automatically. The details of the growth policy are not specified.

This class and its iterator implement all of the optional methods of the **Collection** (p. 1216) and **Iterator** (p. 2222) interfaces. The **Iterator** (p. 2222) provided in method **iterator()** (p. 3121) is not guaranteed to traverse the elements of the priority queue in any particular order. If you need ordered traversal, consider using `Arrays::sort ( pq.toArray() )`.

Note that this implementation is not synchronized. Multiple threads should not access a **PriorityQueue** (p. 3116) instance concurrently if any of the threads modifies the queue. Instead, use the thread-safe `PriorityBlockingQueue` class.

Implementation note: this implementation provides  $O(\log(n))$  time for the enqueueing and dequeuing methods (offer, poll, **remove()** (p. 3123) and add); linear time for the `remove(Object)` and `contains(Object)` methods; and constant time for the retrieval methods (peek, element, and size).

Since

1.0

### 6.625.2 Constructor & Destructor Documentation

**6.625.2.1** `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue ( ) [inline]`

Creates a Priority **Queue** (p. 3239) with the default initial capacity.

**6.625.2.2** `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue ( std::size_t initialCapacity ) [inline]`

Creates a Priority **Queue** (p. 3239) with the capacity value supplied.

**Parameters**

<i>initialCapacity</i>	The initial number of elements allocated to this <b>PriorityQueue</b> (p. 3116).
------------------------	--

**6.625.2.3** `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue ( std::size_t initialCapacity, Comparator< E > * comparator ) [inline]`

Creates a **Priority Queue** (p. 3239) with the default initial capacity.

This new **PriorityQueue** (p. 3116) takes ownership of the passed **Comparator** (p. 1251) instance and uses that to determine the ordering of the elements in the **Queue** (p. 3239).

**Parameters**

<i>initialCapacity</i>	The initial number of elements allocated to this <b>PriorityQueue</b> (p. 3116).
<i>comparator</i>	The <b>Comparator</b> (p. 1251) instance to use in sorting the elements in the <b>Queue</b> (p. 3239).

**Exceptions**

<i>NullPointerException</i>	if the passed <b>Comparator</b> (p. 1251) is NULL.
-----------------------------	--

**6.625.2.4** `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue ( const Collection< E > & source ) [inline]`

Creates a **PriorityQueue** (p. 3116) containing the elements in the specified **Collection** (p. 1216).

**Parameters**

<i>source</i>	the <b>Collection</b> (p. 1216) whose elements are to be placed into this priority queue
---------------	--

**6.625.2.5** `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue ( const PriorityQueue< E > & source ) [inline]`

Creates a **PriorityQueue** (p. 3116) containing the elements in the specified priority queue.

This priority queue will be ordered according to the same ordering as the given priority queue.

**Parameters**

<i>source</i>	the priority queue whose elements are to be placed into this priority queue
---------------	---

6.625.2.6 `template<typename E> virtual decaf::util::PriorityQueue< E  
>::~~PriorityQueue( ) [inline, virtual]`

### 6.625.3 Member Function Documentation

6.625.3.1 `template<typename E> virtual bool decaf::util::PriorityQueue<  
E >::add ( const E & value ) throw (  
lang::exceptions::UnsupportedOperationException,  
lang::exceptions::IllegalArgumentException,  
lang::exceptions::IllegalStateException) [inline, virtual]`

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an `IllegalStateException` if no space is currently available.

This implementation returns true if offer succeeds, else throws an `IllegalStateException`.

#### Parameters

<i>value</i>	- the element to offer to the <b>Queue</b> (p. 3239).
--------------	---

#### Returns

true if the add succeeds.

#### Exceptions

<i>IllegalArgumentException</i>	if the element cannot be added.
---------------------------------	---------------------------------

Reimplemented from `decaf::util::AbstractQueue< E >` (p. 177).

References `DECAF_CATCH_EXCEPTION_CONVERT`, `DECAF_CATCH_RETHROW`, `DECAF_CATCHALL_THROW`, and `decaf::util::PriorityQueue< E >::offer()`.

6.625.3.2 `template<typename E> virtual void decaf::util::PriorityQueue< E >::clear  
( ) throw ( lang::exceptions::UnsupportedOperationException )  
[inline, virtual]`

Removes all elements of the queue.

This implementation repeatedly invokes `poll` until it returns the empty marker.

Reimplemented from `decaf::util::AbstractQueue< E >` (p. 178).

6.625.3.3 `template<typename E> decaf::lang::Pointer< Comparator<E> >  
decaf::util::PriorityQueue< E >::comparator ( ) const [inline]`

obtains a Copy of the `Pointer` instance that this **PriorityQueue** (p. 3116) is using to compare the elements in the queue with.

The returned value is a copy, the caller cannot change the value if the internal Pointer value.

### Returns

a copy of the **Comparator** (p. 1251) Pointer being used by this **Queue** (p. 3239).

```
6.625.3.4  template<typename E> virtual decaf::util::Iterator<E>*
           decaf::util::PriorityQueue< E >::iterator ( ) const  [inline,
           virtual]
```

Implements **decaf::lang::Iterable**< E > (p. 2222).

```
6.625.3.5  template<typename E> virtual decaf::util::Iterator<E>*
           decaf::util::PriorityQueue< E >::iterator ( ) [inline, virtual]
```

### Returns

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable**< E > (p. 2221).

References decaf::util::PriorityQueue< E >::PriorityQueueIterator.

```
6.625.3.6  template<typename E> virtual bool decaf::util::PriorityQueue< E >::offer (
           const E & value ) throw ( decaf::lang::exceptions::NullPointerException,
           decaf::lang::exceptions::IllegalArgumentException ) [inline,
           virtual]
```

Inserts the specified element into the queue provided that the condition allows such an operation.

The method is generally preferable to the collection.add(E), since the latter might throw an exception if the operation fails.

### Parameters

<i>value</i>	the specified element to insert into the queue.
--------------	---

### Returns

true if the operation succeeds and false if it fails.

### Exceptions

<i>NullPointerException</i>	if the <b>Queue</b> (p. 3239) implementation does not allow Null values to be inserted into the <b>Queue</b> (p. 3239).
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this queue

Implements **decaf::util::Queue**< E > (p. 3241).

Referenced by **decaf::util::PriorityQueue**< E >::add().

**6.625.3.7** `template<typename E> PriorityQueue<E>& decaf::util::PriorityQueue<E>::operator=( const PriorityQueue< E > & source ) [inline]`

Assignment operator, assign another **PriorityQueue** (p. 3116) to this one.

#### Parameters

<i>source</i>	The <b>PriorityQueue</b> (p. 3116) to copy to this one.
---------------	---

**6.625.3.8** `template<typename E> PriorityQueue<E>& decaf::util::PriorityQueue<E>::operator=( const Collection< E > & source ) [inline]`

Assignment operator, assign another **Collection** (p. 1216) to this one.

#### Parameters

<i>source</i>	The <b>Collection</b> (p. 1216) to copy to this one.
---------------	--

**6.625.3.9** `template<typename E> virtual bool decaf::util::PriorityQueue< E >::peek ( E & result ) const [inline, virtual]`

Gets but not removes the element in the head of the queue.

The result if successful is assigned to the result parameter.

#### Parameters

<i>result</i>	Reference to an instance of the contained type to assigned the removed value to.
---------------	--

#### Returns

true if the element at the head of the queue was removed and assigned to the result parameter.

Implements **decaf::util::Queue**< E > (p. 3242).

**6.625.3.10** `template<typename E> virtual bool decaf::util::PriorityQueue< E >::poll ( E & result ) [inline, virtual]`

Gets and removes the element in the head of the queue.

If the operation succeeds the value of the element at the head of the **Queue** (p. 3239) is assigned to the result parameter and the method returns true. If the operation fails the method returns false and the value of the result parameter is undefined.

**Parameters**

<i>result</i>	Reference to an instance of the contained type to assigned the removed value to.
---------------	--

**Returns**

true if the element at the head of the queue was removed and assigned to the result parameter.

Implements **decaf::util::Queue< E >** (p. 3242).

**6.625.3.11** `template<typename E> virtual E decaf::util::PriorityQueue< E >::remove  
( ) throw ( decaf::lang::exceptions::NoSuchElementException )  
[inline, virtual]`

Retrieves and removes the head of this queue.

This method differs from poll only in that it throws an exception if this queue is empty.

This implementation returns the result of poll unless the queue is empty.

**Returns**

a copy of the element in the head of the queue.

**Exceptions**

<i>NoSuchElementException</i>	if the queue is empty.
-------------------------------	------------------------

Reimplemented from **decaf::util::AbstractQueue< E >** (p. 179).

**6.625.3.12** `template<typename E> virtual bool decaf::util::PriorityQueue<  
E >::remove ( const E & value ) throw (  
lang::exceptions::UnsupportedOperationException,  
lang::exceptions::IllegalArgumentException ) [inline,  
virtual]`

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method

and this collection contains the specified object.

#### Parameters

<i>value</i>	- element to be removed from this collection, if present
--------------	--

#### Returns

true if an element was removed as a result of this call

#### Exceptions

<i>UnsupportedOperationException</i>	if the remove operation is not supported by this collection.
<i>IllegalArgumentException</i>	If the value is not a valid entry for this <b>Collection</b> (p. 1216).

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 168).

**6.625.3.13** `template<typename E> virtual std::size_t decaf::util::PriorityQueue< E  
>::size ( ) const [inline, virtual]`

Returns the number of elements in this collection.

If this collection contains more than Integer.MAX\_VALUE elements, returns Integer.MAX\_VALUE.

#### Returns

the number of elements in this collection

Implements **decaf::util::Collection**< **E** > (p. 1226).

### 6.625.4 Friends And Related Function Documentation

**6.625.4.1** `template<typename E> friend class PriorityQueueIterator [friend]`

Referenced by `decaf::util::PriorityQueue< E >::iterator()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/ProducerAck.h`

## 6.626 activemq::commands::ProducerAck Class Reference

```
#include <src/main/activemq/commands/ProducerAck.h>
```

Inheritance diagram for `activemq::commands::ProducerAck`:

## Public Member Functions

- **ProducerAck ()**
- virtual **~ProducerAck ()**
- virtual unsigned char **getDataStructureType ()** const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ProducerAck \* cloneDataStructure ()** const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure (const DataStructure \*src)**  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString ()** const  
*Returns a string containing the information for this DataStructure (p. 1713) such as its type and value of its elements.*
- virtual bool **equals (const DataStructure \*value)** const  
*Compares the DataStructure (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer< ProducerId > & getProducerId ()** const
- virtual **Pointer< ProducerId > & getProducerId ()**
- virtual void **setProducerId (const Pointer< ProducerId > &producerId)**
- virtual int **getSize ()** const
- virtual void **setSize (int size)**
- virtual bool **isProducerAck ()** const
- virtual **Pointer< Command > visit (activemq::state::CommandVisitor \*visitor)**  
 throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

## Static Public Attributes

- static const unsigned char **ID\_PRODUCERACK = 19**

## Protected Attributes

- **Pointer< ProducerId > producerId**
- **int size**



### 6.626.1 Constructor & Destructor Documentation

6.626.1.1 `activemq::commands::ProducerAck::ProducerAck ( )`

6.626.1.2 `virtual activemq::commands::ProducerAck::~~ProducerAck ( ) [virtual]`

### 6.626.2 Member Function Documentation

6.626.2.1 `virtual ProducerAck* activemq::commands::ProducerAck::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1714).

6.626.2.2 `virtual void activemq::commands::ProducerAck::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

6.626.2.3 `virtual bool activemq::commands::ProducerAck::equals ( const DataStructure * value ) const [virtual]`

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

**6.626.2.4** `virtual unsigned char activemq::commands::ProducerAck::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataSetructure** (p. 1713) type copy.

Implements **activemq::commands::DataSetructure** (p. 1717).

**6.626.2.5** `virtual const Pointer<ProducerId>& activemq::commands::ProducerAck::getProducerId ( ) const [virtual]`

**6.626.2.6** `virtual Pointer<ProducerId>& activemq::commands::ProducerAck::getProducerId ( ) [virtual]`

**6.626.2.7** `virtual int activemq::commands::ProducerAck::getSize ( ) const [virtual]`

**6.626.2.8** `virtual bool activemq::commands::ProducerAck::isProducerAck ( ) const [inline, virtual]`

#### Returns

an answer of true to the **isProducerAck()** (p. 3127) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 769).

**6.626.2.9** `virtual void activemq::commands::ProducerAck::setProducerId ( const Pointer<ProducerId> & producerId ) [virtual]`

**6.626.2.10** `virtual void activemq::commands::ProducerAck::setSize ( int size ) [virtual]`

**6.626.2.11** `virtual std::string activemq::commands::ProducerAck::toString ( ) const [virtual]`

Returns a string containing the information for this **DataSetructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

## 6.627 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller Class Reference 3131

---

**6.626.2.12** `virtual Pointer<Command> activemq::commands::ProducerAck::visit  
( activemq::state::CommandVisitor * visitor ) throw (  
exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

### Returns

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

### 6.626.3 Field Documentation

**6.626.3.1** `const unsigned char activemq::commands::ProducerAck::ID_-  
PRODUCERACK = 19 [static]`

**6.626.3.2** `Pointer<ProducerId> activemq::commands::ProducerAck::producerId  
[protected]`

**6.626.3.3** `int activemq::commands::ProducerAck::size [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ProducerAck.h`

## 6.627 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3128).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller`:

### Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*

- virtual unsigned char **getDataStructureType** () const

*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.627.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3128). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.627.2 Constructor & Destructor Documentation

- 6.627.2.1 `activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::ProducerAckMarshaller ( ) [inline]`
- 6.627.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::~~ProducerAckMarshaller ( ) [inline, virtual]`

## 6.627.3 Member Function Documentation

- 6.627.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

- 6.627.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

- 6.627.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 808).

**6.627.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 809).

**6.627.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 810).

**6.627.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 811).

**6.627.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 813).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h`

## 6.628 `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3133).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ProducerAckMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller`:

### Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )



*Write a object instance to data output stream.*

### 6.628.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3133). NOTE!  
This file is auto generated - do not modify! if you need to make a change, please see  
the Java Classes in the activemq-openwire-generator module

### 6.628.2 Constructor & Destructor Documentation

**6.628.2.1** `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::ProducerAckMarshaller  
( ) [inline]`

**6.628.2.2** `virtual activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::~~ProducerAckMarshaller  
( ) [inline, virtual]`

### 6.628.3 Member Function Documentation

**6.628.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::createObject ( )  
const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

**6.628.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::getDataStructureType  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.628.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::looseMarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (  
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 780).

**6.628.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 781).

**6.628.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 782).

**6.628.3.6** virtual void activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::tightMarshal2  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure  
 \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
 utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
 [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 783).

**6.628.3.7** virtual void activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::tightUnmarshal  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream  
 \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ProducerAckMarshaller.h`

## 6.629 **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller** Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3137).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller**:

**Public Member Functions**

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.629.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3137). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.629.2 Constructor & Destructor Documentation

**6.629.2.1** **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::ProducerAckMarshaller**  
( ) [inline]

**6.629.2.2** **virtual activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::~~ProducerAckMarshaller**  
( ) [inline, virtual]

### 6.629.3 Member Function Documentation

**6.629.3.1** **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.629.3.2** **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.629.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 772).

**6.629.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 774).

6.629.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 775).

6.629.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 776).

```
6.629.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ProducerAckMarshaller.h**

## 6.630 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3141).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMars
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller**:

#### Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*



- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.630.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3141). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.630.2 Constructor & Destructor Documentation

**6.630.2.1** **activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::ProducerAckMarshaller**  
( ) [inline]

**6.630.2.2** **virtual activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::~~ProducerAckMarshaller**  
( ) [inline, virtual]

### 6.630.3 Member Function Documentation

**6.630.3.1** **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.630.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.630.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 794).

**6.630.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 795).

**6.630.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 796).

**6.630.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 797).

**6.630.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h`

## 6.631 **activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller** Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3145).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ProducerAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller**:

## Public Member Functions

- **ProducerAckMarshaller ()**
- virtual **~ProducerAckMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.631.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3145). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.631.2 Constructor & Destructor Documentation

6.631.2.1 `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::ProducerAckMarshaller ( ) [inline]`

6.631.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::~~ProducerAckMarshaller ( ) [inline, virtual]`

## 6.631.3 Member Function Documentation

6.631.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.631.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.631.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 801).

**6.631.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 802).

**6.631.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 803).

```
6.631.3.6  virtual void activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 804).

```
6.631.3.7  virtual void activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 806).

The documentation for this class was generated from the following file:



- src/main/activemq/wireformat/openwire/marshal/v6/ProducerAckMarshaller.h

## 6.632 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3150).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ProducerAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller:

### Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.632.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3150). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.632.2 Constructor & Destructor Documentation

6.632.2.1 **activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::ProducerAckMarshaller**  
( ) [inline]

6.632.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::~~ProducerAckMarshaller**  
( ) [inline, virtual]

### 6.632.3 Member Function Documentation

6.632.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.632.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.632.3.3 **virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 787).

**6.632.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 788).

**6.632.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 789).

```
6.632.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 790).

```
6.632.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 792).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ProducerAckMarshaller.h**

## 6.633 activemq::cmsutil::ProducerCallback Class Reference

Callback for sending a message to a CMS destination.

```
#include <src/main/activemq/cmsutil/ProducerCallback.h>
```

Inheritance diagram for activemq::cmsutil::ProducerCallback:

### Public Member Functions

- virtual **~ProducerCallback** ()
- virtual void **doInCms** (**cms::Session** \*session, **cms::MessageProducer** \*producer)=0  
throw ( cms::CMSException )  
*Execute an action given a session and producer.*

#### 6.633.1 Detailed Description

Callback for sending a message to a CMS destination.

#### 6.633.2 Constructor & Destructor Documentation

**6.633.2.1** virtual **activemq::cmsutil::ProducerCallback::~ProducerCallback** ( ) [inline, virtual]

#### 6.633.3 Member Function Documentation

**6.633.3.1** virtual void **activemq::cmsutil::ProducerCallback::doInCms** ( **cms::Session** \**session*, **cms::MessageProducer** \* *producer* ) throw ( cms::CMSException )  
[pure virtual]

Execute an action given a session and producer.

### Parameters

<i>session</i>	the CMS Session
<i>producer</i>	the CMS Producer

### Exceptions

<b><i>cms::CMSException</i></b> (p. 1190)	if thrown by CMS API methods
--	------------------------------

Implemented in **activemq::cmsutil::CmsTemplate::SendExecutor** (p. 3446).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/ProducerCallback.h`

## 6.634 activemq::cmsutil::CmsTemplate::ProducerExecutor Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for `activemq::cmsutil::CmsTemplate::ProducerExecutor`:

### Public Member Functions

- **ProducerExecutor** (**ProducerCallback** \*action, **CmsTemplate** \*parent, **cms::Destination** \*destination)
- virtual **~ProducerExecutor** ()
- virtual void **doInCms** (**cms::Session** \*session) throw ( **cms::CMSException** )  
*Execute any number of operations against the supplied CMS session.*
- virtual **cms::Destination** \* **getDestination** (**cms::Session** \*session **AMQCPP\_UNUSED**) throw ( **cms::CMSException** )

### Protected Member Functions

- **ProducerExecutor** (const **ProducerExecutor** &)
- **ProducerExecutor** & **operator=** (const **ProducerExecutor** &)

### Protected Attributes

- **ProducerCallback** \* action
- **CmsTemplate** \* parent
- **cms::Destination** \* destination

## 6.634 activemq::cmsutil::CmsTemplate::ProducerExecutor Class Reference 3159

### 6.634.1 Constructor & Destructor Documentation

6.634.1.1 `activemq::cmsutil::CmsTemplate::ProducerExecutor::ProducerExecutor ( const ProducerExecutor & ) [inline, protected]`

6.634.1.2 `activemq::cmsutil::CmsTemplate::ProducerExecutor::ProducerExecutor ( ProducerCallback * action, CmsTemplate * parent, cms::Destination * destination ) [inline]`

6.634.1.3 `virtual activemq::cmsutil::CmsTemplate::ProducerExecutor::~~ProducerExecutor ( ) [inline, virtual]`

### 6.634.2 Member Function Documentation

6.634.2.1 `virtual void activemq::cmsutil::CmsTemplate::ProducerExecutor::doInCms ( cms::Session * session ) throw ( cms::CMSEException ) [virtual]`

Execute any number of operations against the supplied CMS session.

#### Parameters

<i>session</i>	the CMS Session
----------------	-----------------

#### Exceptions

<i>cms::CMSEException</i> (p. 1190)	if thrown by CMS API methods
--	------------------------------

Implements `activemq::cmsutil::SessionCallback` (p. 3476).

6.634.2.2 `virtual cms::Destination* activemq::cmsutil::CmsTemplate::ProducerExecutor::getDestination ( cms::Session *session AMQCPP_UNUSED ) throw ( cms::CMSEException ) [inline, virtual]`

6.634.2.3 `ProducerExecutor& activemq::cmsutil::CmsTemplate::ProducerExecutor::operator= ( const ProducerExecutor & ) [inline, protected]`

### 6.634.3 Field Documentation

6.634.3.1 `ProducerCallback* activemq::cmsutil::CmsTemplate::ProducerExecutor::action [protected]`

6.634.3.2 `cms::Destination* activemq::cmsutil::CmsTemplate::ProducerExecutor::destination [protected]`

6.634.3.3 `CmsTemplate* activemq::cmsutil::CmsTemplate::ProducerExecutor::parent [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

## 6.635 `activemq::commands::ProducerId` Class Reference

```
#include <src/main/activemq/commands/ProducerId.h>
```

Inheritance diagram for `activemq::commands::ProducerId`:

### Public Types

- `typedef decaf::lang::PointerComparator< ProducerId > COMPARATOR`

### Public Member Functions

- **ProducerId** ()
- **ProducerId** (const **ProducerId** &other)
- **ProducerId** (const **SessionId** &sessionId, long long consumerId)
- **ProducerId** (std::string producerId)
- virtual **~ProducerId** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ProducerId \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- const **Pointer**< **SessionId** > & **getParentId** () const
- void **setProducerSessionKey** (std::string sessionKey)
- virtual const std::string & **getConnectionId** () const
- virtual std::string & **getConnectionId** ()
- virtual void **setConnectionId** (const std::string &connectionId)



- virtual long long **getValue** () const
- virtual void **setValue** (long long **value**)
- virtual long long **getSessionId** () const
- virtual void **setSessionId** (long long **sessionId**)
- virtual int **compareTo** (const **ProducerId** &**value**) const
- virtual bool **equals** (const **ProducerId** &**value**) const
- virtual bool **operator==** (const **ProducerId** &**value**) const
- virtual bool **operator<** (const **ProducerId** &**value**) const
- **ProducerId** & **operator=** (const **ProducerId** &**other**)

### Static Public Attributes

- static const unsigned char **ID\_PRODUCERID** = 123

### Protected Attributes

- std::string **connectionId**
- long long **value**
- long long **sessionId**

## 6.635.1 Member Typedef Documentation

- 6.635.1.1 **typedef decaf::lang::PointerComparator<ProducerId>  
activemq::commands::ProducerId::COMPARATOR**

## 6.635.2 Constructor & Destructor Documentation

- 6.635.2.1 **activemq::commands::ProducerId::ProducerId ( )**
- 6.635.2.2 **activemq::commands::ProducerId::ProducerId ( const **ProducerId** & *other* )**
- 6.635.2.3 **activemq::commands::ProducerId::ProducerId ( const **SessionId** & *sessionId*, long long *consumerId* )**
- 6.635.2.4 **activemq::commands::ProducerId::ProducerId ( std::string *producerId* )**
- 6.635.2.5 **virtual activemq::commands::ProducerId::~~ProducerId ( ) [virtual]**

## 6.635.3 Member Function Documentation

- 6.635.3.1 **virtual **ProducerId**\* activemq::commands::ProducerId::cloneDataStructure ( )  
const [virtual]**

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

**Returns**

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

**6.635.3.2** `virtual int activemq::commands::ProducerId::compareTo ( const ProducerId & value ) const` [virtual]

**6.635.3.3** `virtual void activemq::commands::ProducerId::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

**Parameters**

<i>src</i>	- Source Object
------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1715).

**6.635.3.4** `virtual bool activemq::commands::ProducerId::equals ( const DataStructure * value ) const` [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns**

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1716).

**6.635.3.5** `virtual bool activemq::commands::ProducerId::equals ( const ProducerId & value ) const` [virtual]

**6.635.3.6** `virtual const std::string& activemq::commands::ProducerId::getConnectionId ( ) const` [virtual]

**6.635.3.7** `virtual std::string& activemq::commands::ProducerId::getConnectionId ( )` [virtual]

**6.635.3.8** `virtual unsigned char activemq::commands::ProducerId::getDataStructureType ( ) const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns**

new **DataSet** (p. 1713) type copy.

Implements **activemq::commands::DataSet** (p. 1717).

**6.635.3.9** `const Pointer<SessionId>& activemq::commands::ProducerId::getParentId ( ) const`

**6.635.3.10** `virtual long long activemq::commands::ProducerId::getSessionId ( ) const [virtual]`

**6.635.3.11** `virtual long long activemq::commands::ProducerId::getValue ( ) const [virtual]`

**6.635.3.12** `virtual bool activemq::commands::ProducerId::operator< ( const ProducerId & value ) const [virtual]`

**6.635.3.13** `ProducerId& activemq::commands::ProducerId::operator= ( const ProducerId & other )`

**6.635.3.14** `virtual bool activemq::commands::ProducerId::operator== ( const ProducerId & value ) const [virtual]`

**6.635.3.15** `virtual void activemq::commands::ProducerId::setConnectionId ( const std::string & connectionId ) [virtual]`

**6.635.3.16** `void activemq::commands::ProducerId::setProducerSessionKey ( std::string sessionKey )`

**6.635.3.17** `virtual void activemq::commands::ProducerId::setSessionId ( long long sessionId ) [virtual]`

**6.635.3.18** `virtual void activemq::commands::ProducerId::setValue ( long long value ) [virtual]`

**6.635.3.19** `virtual std::string activemq::commands::ProducerId::toString ( ) const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1713) such as its type and value of its elements.

**Returns**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 841).

### 6.635.4 Field Documentation

6.635.4.1 `std::string activemq::commands::ProducerId::connectionId`  
[protected]

6.635.4.2 `const unsigned char activemq::commands::ProducerId::ID_-  
PRODUCERID = 123` [static]

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

6.635.4.3 `long long activemq::commands::ProducerId::sessionId` [protected]

6.635.4.4 `long long activemq::commands::ProducerId::value` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ProducerId.h`

## 6.636 `activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller` Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3161).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarsh
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller`:

### Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataInputStream \* dataIn**, **utils::BooleanStream \* bs**)  
**throw** ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.636.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3161). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.636.2 Constructor & Destructor Documentation

- 6.636.2.1 **activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::ProducerIdMarshaller**  
( ) [inline]
- 6.636.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::~~ProducerIdMarshaller**  
( ) [inline, virtual]

### 6.636.3 Member Function Documentation

- 6.636.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.636.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.636.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1675).

**6.636.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

## 6.636 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller

### Class Reference 3167

---

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

```
6.636.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.636.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.636.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ProducerIdMarshaller.h**

## 6.637 activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3165).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarsh
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller**:

#### Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*



- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.637.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3165). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.637.2 Constructor & Destructor Documentation

6.637.2.1 **activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::ProducerIdMarshaller**  
( ) [inline]

6.637.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::~~ProducerIdMarshaller**  
( ) [inline, virtual]

### 6.637.3 Member Function Documentation

6.637.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.637.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.637.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.637.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.637.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.637.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.637.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h`

## 6.638 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3169).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ProducerIdMarsh
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller**:

#### Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.638.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3169). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.638.2 Constructor & Destructor Documentation

6.638.2.1 **activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::ProducerIdMarshaller**  
( ) [inline]

6.638.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::~~ProducerIdMarshaller**  
( ) [inline, virtual]

### 6.638.3 Member Function Documentation

6.638.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.638.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.638.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.638.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.638.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.638.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.638.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ProducerIdMarshaller.h`

## 6.639 activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3173).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ProducerIdMarsh
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller**:

#### Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*



- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.639.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3173). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.639.2 Constructor & Destructor Documentation

6.639.2.1 **activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::ProducerIdMarshaller**  
( ) [inline]

6.639.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::~~ProducerIdMarshaller**  
( ) [inline, virtual]

### 6.639.3 Member Function Documentation

6.639.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.639.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.639.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.639.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.639.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.639.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.639.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ProducerIdMarshaller.h`

## 6.640 **activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3177).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ProducerIdMarsh
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller**:

#### Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.640.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3177). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.640.2 Constructor & Destructor Documentation

6.640.2.1 **activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::ProducerIdMarshaller**  
( ) [inline]

6.640.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::~~ProducerIdMarshaller**  
( ) [inline, virtual]

### 6.640.3 Member Function Documentation

6.640.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.640.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.640.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.640.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.640.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.640.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.640.3.7** `virtual void activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ProducerIdMarshaller.h`

## 6.641 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3181).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ProducerIdMarsh
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller**:

#### Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*



- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.641.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3181). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.641.2 Constructor & Destructor Documentation

6.641.2.1 **activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::ProducerIdMarshaller**  
( ) [inline]

6.641.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::~~ProducerIdMarshaller**  
( ) [inline, virtual]

### 6.641.3 Member Function Documentation

6.641.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.641.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.641.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.641.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.641.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

**6.641.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

**6.641.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ProducerIdMarshaller.h`

## 6.642 activemq::commands::ProducerInfo Class Reference

```
#include <src/main/activemq/commands/ProducerInfo.h>
```

Inheritance diagram for `activemq::commands::ProducerInfo`:

#### Public Member Functions

- **ProducerInfo** ()
- virtual **~ProducerInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaller share.*
- virtual **ProducerInfo \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)

*Copy the contents of the passed object into this object's members, overwriting any existing data.*

- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- **Pointer< RemoveInfo > createRemoveCommand** () const
- virtual const **Pointer< ProducerId > &getProducerId** () const
- virtual **Pointer< ProducerId > &getProducerId** ()
- virtual void **setProducerId** (const **Pointer< ProducerId > &producerId**)
- virtual const **Pointer< ActiveMQDestination > &getDestination** () const
- virtual **Pointer< ActiveMQDestination > &getDestination** ()
- virtual void **setDestination** (const **Pointer< ActiveMQDestination > &destination**)
- virtual const std::vector< **decaf::lang::Pointer< BrokerId > > &getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer< BrokerId > > &getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer< BrokerId > > &brokerPath**)
- virtual bool **isDispatchAsync** () const
- virtual void **setDispatchAsync** (bool **dispatchAsync**)
- virtual int **getWindowSize** () const
- virtual void **setWindowSize** (int **windowSize**)
- virtual bool **isProducerInfo** () const
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_PRODUCERINFO** = 6

### Protected Attributes

- **Pointer< ProducerId > producerId**
- **Pointer< ActiveMQDestination > destination**
- std::vector< **decaf::lang::Pointer< BrokerId > > brokerPath**
- bool **dispatchAsync**
- int **windowSize**

## 6.642.1 Constructor & Destructor Documentation

6.642.1.1 `activemq::commands::ProducerInfo::ProducerInfo ( )`

6.642.1.2 `virtual activemq::commands::ProducerInfo::~~ProducerInfo ( ) [virtual]`

## 6.642.2 Member Function Documentation

6.642.2.1 `virtual ProducerInfo* activemq::commands::ProducerInfo::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1714).

6.642.2.2 `virtual void activemq::commands::ProducerInfo::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

6.642.2.3 `Pointer<RemoveInfo> activemq::commands::ProducerInfo::createRemoveCommand ( ) const`

6.642.2.4 `virtual bool activemq::commands::ProducerInfo::equals ( const DataStructure * value ) const [virtual]`

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

- 6.642.2.5 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ProducerInfo::getBrokerPath ( ) const [virtual]`
- 6.642.2.6 `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ProducerInfo::getBrokerPath ( ) [virtual]`
- 6.642.2.7 `virtual unsigned char activemq::commands::ProducerInfo::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataSetructure** (p. 1713) type copy.

Implements **activemq::commands::DataSetructure** (p. 1717).

- 6.642.2.8 `virtual const Pointer<ActiveMQDestination>& activemq::commands::ProducerInfo::getDestination ( ) const [virtual]`
- 6.642.2.9 `virtual Pointer<ActiveMQDestination>& activemq::commands::ProducerInfo::getDestination ( ) [virtual]`
- 6.642.2.10 `virtual Pointer<ProducerId>& activemq::commands::ProducerInfo::getProducerId ( ) [virtual]`
- 6.642.2.11 `virtual const Pointer<ProducerId>& activemq::commands::ProducerInfo::getProducerId ( ) const [virtual]`
- 6.642.2.12 `virtual int activemq::commands::ProducerInfo::getWindowSize ( ) const [virtual]`
- 6.642.2.13 `virtual bool activemq::commands::ProducerInfo::isDispatchAsync ( ) const [virtual]`
- 6.642.2.14 `virtual bool activemq::commands::ProducerInfo::isProducerInfo ( ) const [inline, virtual]`

#### Returns

an answer of true to the **isProducerInfo()** (p. 3188) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 769).

- 6.642.2.15 `virtual void activemq::commands::ProducerInfo::setBrokerPath ( const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath ) [virtual]`
- 6.642.2.16 `virtual void activemq::commands::ProducerInfo::setDestination ( const Pointer< ActiveMQDestination > & destination ) [virtual]`
- 6.642.2.17 `virtual void activemq::commands::ProducerInfo::setDispatchAsync ( bool dispatchAsync ) [virtual]`
- 6.642.2.18 `virtual void activemq::commands::ProducerInfo::setProducerId ( const Pointer< ProducerId > & producerId ) [virtual]`
- 6.642.2.19 `virtual void activemq::commands::ProducerInfo::setWindowSize ( int windowSize ) [virtual]`
- 6.642.2.20 `virtual std::string activemq::commands::ProducerInfo::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

- 6.642.2.21 `virtual Pointer<Command> activemq::commands::ProducerInfo::visit ( activemq::state::CommandVisitor * visitor ) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

### Returns

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).



### 6.642.3 Field Documentation

- 6.642.3.1 `std::vector< decaf::lang::Pointer<BrokerId> >`  
`activemq::commands::ProducerInfo::brokerPath` [protected]
- 6.642.3.2 `Pointer<ActiveMQDestination>` `activemq::commands::ProducerInfo::destination`  
[protected]
- 6.642.3.3 `bool` `activemq::commands::ProducerInfo::dispatchAsync`  
[protected]
- 6.642.3.4 `const unsigned char` `activemq::commands::ProducerInfo::ID_PRODUCERINFO = 6` [static]
- 6.642.3.5 `Pointer<ProducerId>` `activemq::commands::ProducerInfo::producerId`  
[protected]
- 6.642.3.6 `int` `activemq::commands::ProducerInfo::windowSize` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ProducerInfo.h`

## 6.643 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3190).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller`:

### Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual `commands::DataStructure * createObject ()` const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.643.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3190). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.643.2 Constructor & Destructor Documentation

6.643.2.1 **activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::ProducerInfoMarshaller** ( ) [inline]

6.643.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::~~ProducerInfoMarshaller** ( ) [inline, virtual]

### 6.643.3 Member Function Documentation

6.643.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::createObject** ( ) **const** [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.643.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.643.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 780).

**6.643.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 781).

```
6.643.3.5  virtual int activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 782).

```
6.643.3.6  virtual void activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

## 6.644 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller

### Class Reference

3197

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 783).

**6.643.3.7** virtual void **activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ProducerInfoMarshaller.h**

## 6.644 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3194).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller**:

## Public Member Functions

- **ProducerInfoMarshaller ()**
- virtual **~ProducerInfoMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.644.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3194). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.644.2 Constructor & Destructor Documentation

6.644.2.1 `activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::ProducerInfoMarshaller ( ) [inline]`

6.644.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::~~ProducerInfoMarshaller ( ) [inline, virtual]`

## 6.644.3 Member Function Documentation

6.644.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.644.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.644.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 808).

**6.644.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 809).

**6.644.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



## 6.644 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller

### Class Reference 3201

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 810).

```
6.644.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataOutputStream * dataOut,
 utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 811).

```
6.644.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
 dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
 * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 813).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h`

## 6.645 `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3199).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMar
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller`:

### Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.645.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3199). NOTE!  
 This file is auto generated - do not modify! if you need to make a change, please see  
 the Java Classes in the activemq-openwire-generator module

### 6.645.2 Constructor & Destructor Documentation

**6.645.2.1** `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::ProducerInfoMarshaller  
 ( ) [inline]`

**6.645.2.2** `virtual activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::~~ProducerInfoMarshaller  
 ( ) [inline, virtual]`

### 6.645.3 Member Function Documentation

**6.645.3.1** `virtual commands::DataStructure* ac-  
 tivemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::createObject ( )  
 const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.645.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::getDataStructureType  
 ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.645.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::looseMarshal  
 ( OpenWireFormat * wireFormat, commands::DataStructure  
 * dataStructure, decaf::io::DataOutputStream * dataOut ) throw (  
 decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 787).

**6.645.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 788).

**6.645.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 789).

**6.645.3.6** virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::tightMarshal2  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure  
 \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
 utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
 [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 790).

**6.645.3.7** virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::tightUnmarshal  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream  
 \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 792).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ProducerInfoMarshaller.h**

## 6.646 **activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3203).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfoMar
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller**:

**Public Member Functions**

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.646.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3203). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.646.2 Constructor & Destructor Documentation

**6.646.2.1** **activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::ProducerInfoMarshaller**  
( ) [inline]

**6.646.2.2** **virtual activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::~~ProducerInfoMarshaller**  
( ) [inline, virtual]

### 6.646.3 Member Function Documentation

**6.646.3.1** **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.646.3.2** **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.646.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 794).

**6.646.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 795).



6.646.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 796).

6.646.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 797).

```
6.646.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ProducerInfoMarshaller.h**

## 6.647 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3207).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfoMar
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller**:

#### Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.647.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3207). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.647.2 Constructor & Destructor Documentation

**6.647.2.1** **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::ProducerInfoMarshaller**  
( ) [inline]

**6.647.2.2** **virtual activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::~~ProducerInfoMarshaller**  
( ) [inline, virtual]

### 6.647.3 Member Function Documentation

**6.647.3.1** **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.647.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.647.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 772).

**6.647.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 774).

**6.647.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 775).

**6.647.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 776).

```
6.647.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ProducerInfoMarshaller.h**

## 6.648 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3211).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ProducerInfoMar
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller**:

## Public Member Functions

- **ProducerInfoMarshaller ()**
- virtual **~ProducerInfoMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.648.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3211). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.648.2 Constructor & Destructor Documentation

6.648.2.1 `activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::ProducerInfoMarshaller ( ) [inline]`

6.648.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::~~ProducerInfoMarshaller ( ) [inline, virtual]`

## 6.648.3 Member Function Documentation

6.648.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.648.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.648.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 801).

**6.648.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 802).

**6.648.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 803).

```
6.648.3.6 virtual void activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 804).

```
6.648.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 806).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ProducerInfoMarshaller.h`

## 6.649 activemq::state::ProducerState Class Reference

```
#include <src/main/activemq/state/ProducerState.h>
```

### Public Member Functions

- **ProducerState** (const **Pointer**< **ProducerInfo** > &info)
- virtual **~ProducerState** ()
- `std::string toString` () const
- const **Pointer**< **ProducerInfo** > &**getInfo** () const
- void **setTransactionState** (const **Pointer**< **TransactionState** > &transactionState)
- **Pointer**< **TransactionState** > **getTransactionState** () const

### 6.649.1 Constructor & Destructor Documentation

6.649.1.1 **activemq::state::ProducerState::ProducerState** ( const **Pointer**< **ProducerInfo** > & *info* )

6.649.1.2 virtual **activemq::state::ProducerState::~~ProducerState** ( ) [virtual]

### 6.649.2 Member Function Documentation

6.649.2.1 const **Pointer**<**ProducerInfo**>& **activemq::state::ProducerState::getInfo** ( ) const [inline]

6.649.2.2 **Pointer**<**TransactionState**> **activemq::state::ProducerState::getTransactionState** ( ) const

6.649.2.3 void **activemq::state::ProducerState::setTransactionState** ( const **Pointer**< **TransactionState** > & *transactionState* )

6.649.2.4 `std::string` **activemq::state::ProducerState::toString** ( ) const

The documentation for this class was generated from the following file:

- `src/main/activemq/state/ProducerState.h`

## 6.650 decaf::util::Properties Class Reference

Java-like properties class for mapping string names to string values.

```
#include <src/main/decaf/util/Properties.h>
```

## Public Member Functions

- **Properties** ()
- **Properties** (const **Properties** &src)
- virtual ~**Properties** ()
- **Properties** & **operator=** (const **Properties** &src)  
*Assignment Operator.*
- bool **isEmpty** () const  
*Returns true if the properties object is empty.*
- std::size\_t **size** () const
- const char \* **getProperty** (const std::string &name) const  
*Looks up the value for the given property.*
- std::string **getProperty** (const std::string &name, const std::string &defaultValue) const  
*Looks up the value for the given property.*
- std::string **setProperty** (const std::string &name, const std::string &value)  
*Sets the value for a given property.*
- bool **hasProperty** (const std::string &name) const  
*Check to see if the Property exists in the set.*
- std::string **remove** (const std::string &name)  
*Removes the property with the given name.*
- std::vector< std::string > **propertyNames** () const  
*Returns an enumeration of all the keys in this property list, including distinct keys in the default property list if a key of the same name has not already been found from the main properties list.*
- std::vector< std::pair< std::string, std::string > > **toArray** () const  
*Method that serializes the contents of the property map to an array.*
- void **copy** (const **Properties** &source)  
*Copies the contents of the given properties object to this one, if the given **Properties** (p. 3216) instance is NULL then this **List** (p. 2409) is not modified.*
- **Properties** \* **clone** () const  
*Clones this object.*
- void **clear** ()

*Clears all properties from the map.*

- **bool equals** (const **Properties** &source) const  
*Test whether two **Properties** (p. 3216) objects are equivalent.*
- **std::string toString** () const  
*Formats the contents of the **Properties** (p. 3216) Object into a string that can be logged, etc.*
- **void load** (decaf::io::InputStream \*stream) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException )  
*Reads a property list (key and element pairs) from the input byte stream.*
- **void load** (decaf::io::Reader \*reader) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException )  
*Reads a property list (key and element pairs) from the input character stream in a simple line-oriented format.*
- **void store** (decaf::io::OutputStream \*out, const std::string &comment) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )  
*Writes this property list (key and element pairs) in this **Properties** (p. 3216) table to the output stream in a format suitable for loading into a **Properties** (p. 3216) table using the load(InputStream) method.*
- **void store** (decaf::io::Writer \*writer, const std::string &comments) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )  
*Writes this property list (key and element pairs) in this **Properties** (p. 3216) table to the output character stream in a format that can be read by the load(Reader) method.*

## Protected Attributes

- **decaf::lang::Pointer< Properties > defaults**  
*Default list used to answer for any keys not found in the properties list, can be filled in by another implementation of this class.*

### 6.650.1 Detailed Description

Java-like properties class for mapping string names to string values. The **Properties** (p. 3216) list contains a key value pair of properties that can be loaded and stored to a stream. Each **Properties** (p. 3216) instance can contain an internal **Properties** (p. 3216) list that contains default values for keys not found in the **Properties** (p. 3216) **List** (p. 2409).

The **Properties** (p. 3216) list if a Thread Safe class, it can be shared amongst objects in multiple threads without the need for additional synchronization.

**Since**

1.0

**6.650.2 Constructor & Destructor Documentation****6.650.2.1** `decaf::util::Properties::Properties ( )`**6.650.2.2** `decaf::util::Properties::Properties ( const Properties & src )`**6.650.2.3** `virtual decaf::util::Properties::~~Properties ( )` `[virtual]`**6.650.3 Member Function Documentation****6.650.3.1** `void decaf::util::Properties::clear ( )`

Clears all properties from the map.

**6.650.3.2** `Properties* decaf::util::Properties::clone ( ) const`

Clones this object.

**Returns**

a replica of this object.

**6.650.3.3** `void decaf::util::Properties::copy ( const Properties & source )`Copies the contents of the given properties object to this one, if the given **Properties** (p. 3216) instance is NULL then this **List** (p. 2409) is not modified.**Parameters**

<i>source</i>	The source properties object.
---------------	-------------------------------

**6.650.3.4** `bool decaf::util::Properties::equals ( const Properties & source ) const`Test whether two **Properties** (p. 3216) objects are equivalent.Two **Properties** (p. 3216) Objects are considered equivalent when they each contain the same number of elements and each key / value pair contained within the two are equal.

This comparison does not check the contents of the Defaults instance.

**Parameters**

<i>source</i>	The <b>Properties</b> (p. 3216) object to compare this instance to.
---------------	---

**Returns**

true if the contents of the two **Properties** (p. 3216) objects are the same.

**6.650.3.5** `const char* decaf::util::Properties::getProperty ( const std::string & name ) const`

Looks up the value for the given property.

**Parameters**

<i>name</i>	The name of the property to be looked up.
-------------	---

**Returns**

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

**6.650.3.6** `std::string decaf::util::Properties::getProperty ( const std::string & name, const std::string & defaultValue ) const`

Looks up the value for the given property.

**Parameters**

<i>name</i>	The name of the property to be looked up.
<i>defaultValue</i>	The value to be returned if the given property does not exist.

**Returns**

The value of the property specified by *name*, if it exists, otherwise the *defaultValue*.

**6.650.3.7** `bool decaf::util::Properties::hasProperty ( const std::string & name ) const`

Check to see if the Property exists in the set.

**Parameters**

<i>name</i>	The property name to check for in this properties set.
-------------	--

**Returns**

true if property exists, false otherwise.

**6.650.3.8** `bool decaf::util::Properties::isEmpty ( ) const`

Returns true if the properties object is empty.

**Returns**

true if empty

**6.650.3.9** void decaf::util::Properties::load ( decaf::io::InputStream  
\* *stream* ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::IllegalArgumentException,  
decaf::lang::exceptions::NullPointerException )

Reads a property list (key and element pairs) from the input byte stream.

The input stream is in a simple line-oriented format as specified in load(Reader) and is assumed to use the ISO 8859-1 character encoding.

This method does not close the stream upon its return.

**Parameters**

<i>stream</i>	The stream to read the properties data from.
---------------	--

**Exceptions**

<i>IOException</i>	if there is an error while reading from the stream.
<i>IllegalArgumentException</i>	if malformed data is found while reading the properties.
<i>NullPointerException</i>	if the passed stream is Null.

**6.650.3.10** void decaf::util::Properties::load ( decaf::io::Reader  
\* *reader* ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::IllegalArgumentException,  
decaf::lang::exceptions::NullPointerException )

Reads a property list (key and element pairs) from the input character stream in a simple line-oriented format.

**Properties** (p. 3216) are processed in terms of lines. There are two kinds of line, natural lines and logical lines. A natural line is defined as a line of characters that is terminated either by a set of line terminator characters (

or or

) or by the end of the stream. A natural line may be either a blank line, a comment line, or hold all or some of a key-element pair. A logical line holds all the data of a key-element pair, which may be spread out across several adjacent natural lines by escaping the line terminator sequence with a backslash character \. Note that a comment line cannot be extended in this manner; every natural line that is a comment must have its own comment indicator, as described below. Lines are read from input until the end of the stream is reached.

A natural line that contains only white space characters is considered blank and is



ignored. A comment line has an ASCII '#' or '!' as its first non-white space character; comment lines are also ignored and do not encode key-element information. In addition to line terminators, this format considers the characters space (' '), tab (""), and form feed ("") to be white space.

If a logical line is spread across several natural lines, the backslash escaping the line terminator sequence, the line terminator sequence, and any white space at the start of the following line have no effect on the key or element values. The remainder of the discussion of key and element parsing (when loading) will assume all the characters constituting the key and element appear on a single natural line after line continuation characters have been removed. Note that it is not sufficient to only examine the character preceding a line terminator sequence to decide if the line terminator is escaped; there must be an odd number of contiguous backslashes for the line terminator to be escaped. Since the input is processed from left to right, a non-zero even number of 2n contiguous backslashes before a line terminator (or elsewhere) encodes n backslashes after escape processing.

The key contains all of the characters in the line starting with the first non-white space character and up to, but not including, the first unescaped '=', ':', or white space character other than a line terminator. All of these key termination characters may be included in the key by escaping them with a preceding backslash character; for example,

```
\:\=
```

would be the two-character key ":\=" . Line terminator characters can be included using and

escape sequences. Any white space after the key is skipped; if the first non-white space character after the key is '=' or ':', then it is ignored and any white space characters after it are also skipped. All remaining characters on the line become part of the associated element string; if there are no remaining characters, the element is the empty string "". Once the raw character sequences constituting the key and element are identified, escape processing is performed as described above.

As an example, each of the following three lines specifies the key "Truth" and the associated element value "Beauty":

```
Truth = Beauty Truth:Beauty Truth :Beauty
```

As another example, the following three lines specify a single property:

```
fruits apple, banana, pear, \ cantaloupe, watermelon, \ kiwi, mango
```

The key is "fruits" and the associated element is: "apple, banana, pear, cantaloupe, watermelon, kiwi, mango"

Note that a space appears before each \ so that a space will appear after each comma in the final result; the \, line terminator, and leading white space on the continuation line are merely discarded and are not replaced by one or more other characters.

As a third example, the line:

```
cheeses
```

specifies that the key is "cheeses" and the associated element is the empty string "".

Characters in keys and elements can be represented in escape sequences similar to those

used for character and string literals (see §3.3 and §3.10.6 of the Java Language Specification). The differences from the character escape sequences and Unicode escapes used for characters and strings are:

- Octal escapes are not recognized.
- The character sequence **does** not represent a backspace character.
- The method does not treat a backslash character, `\`, before a non-valid escape character as an error; the backslash is silently dropped. For example, in a C++ string the sequence `"\z"` would cause a compile time error. In contrast, this method silently drops the backslash. Therefore, this method treats the two character sequence `"\b"` as equivalent to the single character `'b'`.
- Escapes are not necessary for single and double quotes; however, by the rule above, single and double quote characters preceded by a backslash still yield single and double quote characters, respectively.

This method does not close the Reader upon its return.

#### Parameters

<i>reader</i>	The Reader that provides an character stream as input.
---------------	--

#### Exceptions

<i>IOException</i>	if there is an error while reading from the stream.
<i>IllegalArgumentException</i>	if malformed data is found while reading the properties.
<i>NullPointerException</i>	if the passed stream is Null.

#### 6.650.3.11 Properties& decaf::util::Properties::operator= ( const Properties & src )

Assignment Operator.

#### Parameters

<i>src</i>	The <b>Properties</b> (p. 3216) list to copy to this <b>List</b> (p. 2409).
------------	---

#### Returns

a reference to this **List** (p. 2409) for use in chaining.

#### 6.650.3.12 std::vector<std::string> decaf::util::Properties::propertyNames ( ) const

Returns an enumeration of all the keys in this property list, including distinct keys in the default property list if a key of the same name has not already been found from the main properties list.

**Returns**

a set of keys in this property list where the key and its corresponding value are strings, including the keys in the default property list.

**6.650.3.13** `std::string decaf::util::Properties::remove ( const std::string & name )`

Removes the property with the given name.

**Parameters**

<i>name</i>	The name of the property to remove.
-------------	-------------------------------------

**Returns**

the previous value of the property if set, or empty string.

**6.650.3.14** `std::string decaf::util::Properties::setProperty ( const std::string & name, const std::string & value )`

Sets the value for a given property.

If the property already exists, overwrites the value.

**Parameters**

<i>name</i>	The name of the value to be written.
<i>value</i>	The value to be written.

**Returns**

the old value of the property or empty string if not set.

**6.650.3.15** `std::size_t decaf::util::Properties::size ( ) const`**Returns**

The number of **Properties** (p. 3216) in this **Properties** (p. 3216) Object.

**6.650.3.16** `void decaf::util::Properties::store ( decaf::io::OutputStream * out, const std::string & comment ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )`

Writes this property list (key and element pairs) in this **Properties** (p. 3216) table to the output stream in a format suitable for loading into a **Properties** (p. 3216) table using the load(InputStream) method.

**Properties** (p. 3216) from the defaults table of this **Properties** (p. 3216) table (if any) are not written out by this method.

This method outputs the comments, properties keys and values in the same format as specified in `store(Writer)`, with the following differences:

- The stream is written using the ISO 8859-1 character encoding.
- Characters not in Latin-1 in the comments are written as for their appropriate unicode hexadecimal value `xxxx`.
- Characters less than and characters greater than in property keys or values are written as for the appropriate hexadecimal value `xxxx`.

After the entries have been written, the output stream is flushed. The output stream remains open after this method returns.

#### Parameters

<i>out</i>	The <code>OutputStream</code> instance to write the properties to.
<i>comment</i>	A description of these properties that is written to the output stream.

#### Exceptions

<i>IOException</i>	if there is an error while writing from the stream.
<i>NullPointerException</i>	if the passed stream is <code>Null</code> .

**6.650.3.17** `void decaf::util::Properties::store ( decaf::io::Writer * writer,  
const std::string & comments ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::NullPointerException )`

Writes this property list (key and element pairs) in this **Properties** (p. 3216) table to the output character stream in a format that can be read by the `load(Reader)` method.

**Properties** (p. 3216) from the defaults table of this **Properties** (p. 3216) table (if any) are not written out by this method.

If the `comments` argument is not empty, then an ASCII `#` character, the comments string, and a line separator are first written to the output stream. Thus, the comments can serve as an identifying comment. Any one of a line feed (‘

’), a carriage return (‘’), or a carriage return followed immediately by a line feed in comments is replaced by a line separator generated by the `Writer` and if the next character in comments is not character `#` or character `!` then an ASCII `#` is written out after that line separator.

Next, a comment line is always written, consisting of an ASCII `#` character, the current date and time (as if produced by the `toString` method of **Date** (p. 1719) for the current time), and a line separator as generated by the `Writer`.

Then every entry in this **Properties** (p. 3216) table is written out, one per line. For each entry the key string is written, then an ASCII `=`, then the associated element string. For the key, all space characters are written with a preceding `\` character. For the element, leading space characters, but not embedded or trailing space characters, are written

with a preceding `\` character. The key and element characters `#`, `!`, `=`, and `:` are written with a preceding backslash to ensure that they are properly loaded.

After the entries have been written, the output stream is flushed. The output stream remains open after this method returns.

### Parameters

<i>writer</i>	The Writer instance to use to output the properties.
<i>comments</i>	A description of these properties that is written before writing the properties.

### Exceptions

<i>IOException</i>	if there is an error while writing from the stream.
<i>NullPointerException</i>	if the passed stream is Null.

**6.650.3.18** `std::vector< std::pair< std::string, std::string > > decaf::util::Properties::toArray ( ) const`

Method that serializes the contents of the property map to an array.

### Returns

list of pairs where the first is the name and the second is the value.

**6.650.3.19** `std::string decaf::util::Properties::toString ( ) const`

Formats the contents of the **Properties** (p. 3216) Object into a string that can be logged, etc.

### Returns

string value of this object.

## 6.650.4 Field Documentation

**6.650.4.1** `decaf::lang::Pointer<Properties> decaf::util::Properties::defaults [protected]`

Default list used to answer for any keys not found in the properties list, can be filled in by another implementation of this class.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Properties.h`

## 6.651 decaf::util::logging::PropertiesChangeListener Class Reference

Defines the interface that classes can use to listen for change events on **Properties** (p. 3216).

```
#include <src/main/decaf/util/logging/PropertiesChangeListener.h>
```

### Public Member Functions

- virtual `~PropertiesChangeListener ()`
- virtual void `onPropertiesReset ()=0`  
*Indicates that the **Properties** (p. 3216) have all been reset and should be considered to be back to their default values.*
- virtual void `onPropertyChanged (const std::string &name, const std::string &old-Value, const std::string &newValue)=0`  
*Change Event, called when a property is changed, includes the name of the property that was changed along with it old and new values.*

### 6.651.1 Detailed Description

Defines the interface that classes can use to listen for change events on **Properties** (p. 3216).

Since

1.0

### 6.651.2 Constructor & Destructor Documentation

6.651.2.1 virtual `decaf::util::logging::PropertiesChangeListener::~~PropertiesChangeListener ( )` [inline, virtual]

### 6.651.3 Member Function Documentation

6.651.3.1 virtual void `decaf::util::logging::PropertiesChangeListener::onPropertiesReset ( )` [pure virtual]

Indicates that the **Properties** (p. 3216) have all been reset and should be considered to be back to their default values.

6.651.3.2 virtual void decaf::util::logging::PropertiesChangeListener::onPropertyChanged ( const std::string & *name*, const std::string & *oldValue*, const std::string & *newValue* )  
[pure virtual]

Change Event, called when a property is changed, includes the name of the property that was changed along with it old and new values.

#### Parameters

<i>name</i>	The name of the Property that changed.
<i>oldValue</i>	The old Value of the Property.
<i>newValue</i>	The new Value of the Property.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**PropertiesChangeListener.h**

## 6.652 decaf::net::ProtocolException Class Reference

```
#include <src/main/decaf/net/ProtocolException.h>
```

Inheritance diagram for decaf::net::ProtocolException:

#### Public Member Functions

- **ProtocolException** () throw ()  
*Default Constructor.*
- **ProtocolException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **ProtocolException** (const **ProtocolException** &ex) throw ()  
*Copy Constructor.*
- **ProtocolException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **ProtocolException** (const std::exception \*cause) throw ()  
*Constructor.*
- **ProtocolException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*

- virtual **ProtocolException** \* **clone** () const  
*Clones this exception.*
- virtual ~**ProtocolException** () throw ()

### 6.652.1 Constructor & Destructor Documentation

#### 6.652.1.1 decaf::net::ProtocolException::ProtocolException ( ) throw () [inline]

Default Constructor.

#### 6.652.1.2 decaf::net::ProtocolException::ProtocolException ( const Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

##### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

#### 6.652.1.3 decaf::net::ProtocolException::ProtocolException ( const ProtocolException & ex ) throw () [inline]

Copy Constructor.

##### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

#### 6.652.1.4 decaf::net::ProtocolException::ProtocolException ( const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.  
Sets the message to report, using an optional list of arguments to parse into the message

##### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message



**6.652.1.5** `decaf::net::ProtocolException::ProtocolException ( const std::exception * cause )  
throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.652.1.6** `decaf::net::ProtocolException::ProtocolException ( const char * file, const int  
lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.652.1.7** `virtual decaf::net::ProtocolException::~~ProtocolException ( ) throw ()  
[inline, virtual]`

## 6.652.2 Member Function Documentation

**6.652.2.1** `virtual ProtocolException* decaf::net::ProtocolException::clone ( ) const  
[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 2212).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ProtocolException.h`

## 6.653 decaf::security::PublicKey Class Reference

A public key.

```
#include <src/main/decaf/security/PublicKey.h>
```

Inheritance diagram for decaf::security::PublicKey:

### Public Member Functions

- virtual  $\sim$ **PublicKey** ()

#### 6.653.1 Detailed Description

A public key. This interface contains no methods or constants. It merely serves to group (and provide type safety for) all public key interfaces.

#### 6.653.2 Constructor & Destructor Documentation

**6.653.2.1** virtual decaf::security::PublicKey::~~PublicKey ( ) [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/security/**PublicKey.h**

## 6.654 decaf::io::PushbackInputStream Class Reference

A **PushbackInputStream** (p. 3231) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.

```
#include <src/main/decaf/io/PushbackInputStream.h>
```

Inheritance diagram for decaf::io::PushbackInputStream:

### Public Member Functions

- **PushbackInputStream** (**InputStream** \*stream, bool own=false)  
*Creates a **PushbackInputStream** (p. 3231) and saves its argument, the input stream in, for later use.*
- **PushbackInputStream** (**InputStream** \*stream, int bufSize, bool own=false)  
throw ( decaf::lang::exceptions::IllegalArgumentException )

Creates a **PushbackInputStream** (p. 3231) and saves its argument, the input stream in, for later use.

- virtual **~PushbackInputStream** ()
- void **unread** (unsigned char value) throw ( decaf::io::IOException )  
*Pushes back the given byte, the byte is copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back byte.*
- void **unread** (const unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.*

- void **unread** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.*

- virtual int **available** () const throw ( decaf::io::IOException )

*Indicates the number of bytes available.*

*The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.*

*The default implementation of this method returns zero.*

#### Returns

*the number of bytes available on this input stream.*

#### Exceptions

<b>IOException</b> (p. 2210)	if an I/O error occurs.
------------------------------	-------------------------

- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Skips over and discards n bytes of data from this input stream.*

*The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.*

*The skip method of **InputStream** (p. 2105) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*

#### Parameters

num	The number of bytes to skip.
-----	------------------------------

#### Returns

*total bytes skipped*

*Exceptions*

<b>IOException</b> (p. 2210)	<i>if an I/O error occurs.</i>
UnsupportedOperationException	<i>if the concrete stream class does not support skipping bytes.</i>

- virtual void **mark** (int readLimit)  
*Does nothing except throw an **IOException** (p. 2210).*
- virtual void **reset** () throw ( decaf::io::IOException )  
*Does nothing except throw an **IOException** (p. 2210).*
- virtual bool **markSupported** () const  
*Does nothing except throw an **IOException** (p. 2210).*

**Protected Member Functions**

- virtual int **doReadByte** () throw ( decaf::io::IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

**6.654.1 Detailed Description**

A **PushbackInputStream** (p. 3231) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte. This is useful in situations where it is convenient for a fragment of code to read an indefinite number of data bytes that are delimited by a particular byte value; after reading the terminating byte, the code fragment can "unread" it, so that the next read operation on the input stream will reread the byte that was pushed back. For example, bytes representing the characters constituting an identifier might be terminated by a byte representing an operator character; a method whose job is to read just an identifier can read until it sees the operator and then push the operator back to be re-read.

**Since**

1.0

**6.654.2 Constructor & Destructor Documentation****6.654.2.1 decaf::io::PushbackInputStream::PushbackInputStream ( InputStream \* stream, bool own = false )**

Creates a **PushbackInputStream** (p. 3231) and saves its argument, the input stream in, for later use.

Initially, there is no pushed-back byte.

**Parameters**

<i>stream</i>	The <b>InputStream</b> (p. 2105) instance to wrap.
<i>Boolean</i>	value indicating if this <b>FilterInputStream</b> (p. 1950) owns the wrapped stream.

**6.654.2.2** decaf::io::PushbackInputStream::PushbackInputStream ( **InputStream** \* *stream*, int *bufSize*, bool *own* = false ) throw ( decaf::lang::exceptions::IllegalArgumentException )

Creates a **PushbackInputStream** (p. 3231) and saves its argument, the input stream in, for later use.

Initially, there is no pushed-back byte.

**Parameters**

<i>stream</i>	The <b>InputStream</b> (p. 2105) instance to wrap.
<i>bufSize</i>	The number of byte to allocate for pushback into this stream.
<i>Boolean</i>	value indicating if this <b>FilterInputStream</b> (p. 1950) owns the wrapped stream.

**Exceptions**

<i>IllegalArgumentException</i>	if the bufSize argument is < zero.
---------------------------------	------------------------------------

**6.654.2.3** virtual decaf::io::PushbackInputStream::~~PushbackInputStream ( ) [virtual]

**6.654.3 Member Function Documentation**

**6.654.3.1** virtual int decaf::io::PushbackInputStream::available ( ) const throw ( decaf::io::IOException ) [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

**Returns**

the number of bytes available on this input stream.

**Exceptions**

<i>IOException</i> (p. 2210)	if an I/O error occurs.
---------------------------------	-------------------------

Returns the sum of the number of pushed back bytes if any and the amount of bytes available in the underlying stream via a call to `available`.

Reimplemented from **decaf::io::FilterInputStream** (p. 1953).

```
6.654.3.2  virtual int decaf::io::PushbackInputStream::doReadArrayBounded ( unsigned char
* buffer, int size, int offset, int length ) throw ( decaf::io::IOException,
decaf::lang::exceptions::IndexOutOfBoundsException,
decaf::lang::exceptions::NullPointerException ) [protected,
virtual]
```

Reimplemented from **decaf::io::FilterInputStream** (p. 1954).

```
6.654.3.3  virtual int decaf::io::PushbackInputStream::doReadByte ( ) throw (
decaf::io::IOException ) [protected, virtual]
```

Reimplemented from **decaf::io::FilterInputStream** (p. 1954).

```
6.654.3.4  virtual void decaf::io::PushbackInputStream::mark ( int readLimit ) [virtual]
```

Does nothing except throw an **IOException** (p. 2210).

Marks the current position in the stream A subsequent call to the `reset` method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the `reset` method is called so long the `readLimit` is not reached.

Calling `mark` on a closed stream instance should have no effect.

The default implementation of this method does nothing.

#### Parameters

<i>readLimit</i>	The max bytes read before marked position is invalid.
------------------	---

Reimplemented from **decaf::io::FilterInputStream** (p. 1955).

```
6.654.3.5  virtual bool decaf::io::PushbackInputStream::markSupported ( ) const
[inline, virtual]
```

Does nothing except throw an **IOException** (p. 2210).

Determines if this input stream supports the `mark` and `reset` methods.

Whether or not `mark` and `reset` are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

**Returns**

true if this stream instance supports marks

Reimplemented from **decaf::io::FilterInputStream** (p. 1955).

**6.654.3.6** `virtual void decaf::io::PushbackInputStream::reset ( ) throw ( decaf::io::IOException )` [virtual]

Does nothing except throw an **IOException** (p. 2210).

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2210) might be thrown. \* If such an **IOException** (p. 2210) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2210). \* If an **IOException** (p. 2210) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2210).

**Exceptions**

<b>IOException</b> (p. 2210)	if an I/O error occurs.
---------------------------------	-------------------------

Reimplemented from **decaf::io::FilterInputStream** (p. 1956).

**6.654.3.7** `virtual long long decaf::io::PushbackInputStream::skip ( long long num ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )` [virtual]

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2105) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached.

Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

#### Returns

total bytes skipped

#### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
<b><i>UnsupportedOperationException</i></b>	if the concrete stream class does not support skipping bytes.

This method first skips bytes in the local pushed back buffer before attempting to complete the request by calling the underlying stream skip method with the remainder of bytes that needs to be skipped.

Reimplemented from **decaf::io::FilterInputStream** (p. 1956).

**6.654.3.8** void decaf::io::PushbackInputStream::unread ( const unsigned char \* *buffer*, int *size* ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.

#### Parameters

<i>buffer</i>	The bytes to copy to the front of push back buffer.
<i>size</i>	The size of the array to be copied.

#### Exceptions

<b><i>NullPointerException</i></b>	if the buffer passed is NULL.
<b><i>IndexOutOfBoundsException</i></b>	if the size value given is negative.
<b><i>IOException</i></b> (p. 2210)	if there is not enough space in the pushback buffer or this stream has already been closed.



**6.654.3.9** void decaf::io::PushbackInputStream::unread ( const unsigned char \*  
*buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::NullPointerException )

Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.

#### Parameters

<i>buffer</i>	The bytes to copy to the front of push back buffer.
<i>size</i>	The size of the array to be copied.
<i>offset</i>	The position in the buffer to start copying from.
<i>length</i>	The number of bytes to push back from the passed buffer.

#### Exceptions

<i>NullPointerException</i>	if the buffer passed is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length is greater than the buffer size.
<i>IOException</i> (p. 2210)	if there is not enough space in the pushback buffer or this stream has already been closed.

**6.654.3.10** void decaf::io::PushbackInputStream::unread ( unsigned char *value* ) throw ( decaf::io::IOException )

Pushes back the given byte, the byte is copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back byte.

#### Parameters

<i>value</i>	The byte that is to be placed at the front of the push back buffer.
--------------	---

#### Exceptions

<i>IOException</i> (p. 2210)	if there is not enough space in the pushback buffer or this stream has already been closed.
---------------------------------	---

The documentation for this class was generated from the following file:

- src/main/decaf/io/**PushbackInputStream.h**

## 6.655 cms::Queue Class Reference

An interface encapsulating a provider-specific queue name.

```
#include <src/main/cms/Queue.h>
```

Inheritance diagram for cms::Queue:

## Public Member Functions

- virtual `~Queue()`
- virtual `std::string getQueueName() const` `=0` throw ( `CMSEException` )  
*Gets the name of this queue.*

### 6.655.1 Detailed Description

An interface encapsulating a provider-specific queue name. Messages sent to a **Queue** (p. 3238) are sent to a Single Subscriber on that **Queue** (p. 3238) **Destination** (p. 1776). This allows for Queues to be used as load balances implementing a SEDA based architecture. The length of time that a Provider will store a **Message** (p. 2614) in a **Queue** (p. 3238) is not defined by the CMS API, consult your Provider documentation for this information.

#### Since

1.0

### 6.655.2 Constructor & Destructor Documentation

6.655.2.1 `virtual cms::Queue::~~Queue()` `[inline, virtual]`

### 6.655.3 Member Function Documentation

6.655.3.1 `virtual std::string cms::Queue::getQueueName()` `const` throw ( `CMSEException` )  
`[pure virtual]`

Gets the name of this queue.

#### Returns

The queue name.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- If an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQQueue` (p. 483).

The documentation for this class was generated from the following file:

- src/main/cms/Queue.h

## 6.656 decaf::util::Queue< E > Class Template Reference

A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.

```
#include <src/main/decaf/util/Queue.h>
```

Inheritance diagram for decaf::util::Queue< E >:

### Public Member Functions

- virtual **~Queue** ()
- virtual bool **offer** (const E &value)=0 throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Inserts the specified element into the queue provided that the condition allows such an operation.*
- virtual bool **poll** (E &result)=0  
*Gets and removes the element in the head of the queue.*
- virtual E **remove** ()=0 throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets and removes the element in the head of the queue.*
- virtual bool **peek** (E &result) const =0  
*Gets but not removes the element in the head of the queue.*
- virtual E **element** () const =0 throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets but not removes the element in the head of the queue.*

### 6.656.1 Detailed Description

```
template<typename E> class decaf::util::Queue< E >
```

A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection. Generally, a queue orders its elements by means of first-in-first-out. While priority queue orders its elements according to a comparator specified or the elements' natural order. Furthermore, a stack orders its elements last-in-first out.

**Queue** (p. 3239) does not provide blocking queue methods, which will block until the operation of the method is allowed. `BlockingQueue` interface defines such methods.

Unlike the Java **Queue** (p. 3239) interface the methods of this class cannot return null to indicate that a **Queue** (p. 3239) is empty since null has no meaning for elements such as classes, structs and primitive types and cannot be used in a meaningful way to check for an empty queue. Methods that would have returned null in the Java **Queue** (p. 3239) interface have been altered to return a boolean value indicating if the operation succeeded and take single argument that is a reference to the location where the returned value is to be assigned. This implies that elements in the **Queue** (p. 3239) must be *assignable* in order to utilize these methods.

### Since

1.0

## 6.656.2 Constructor & Destructor Documentation

**6.656.2.1** `template<typename E> virtual decaf::util::Queue< E>::~Queue ( )`  
[inline, virtual]

## 6.656.3 Member Function Documentation

**6.656.3.1** `template<typename E> virtual E decaf::util::Queue< E>::element ( ) const`  
`throw ( decaf::lang::exceptions::NoSuchElementException )` [pure  
virtual]

Gets but not removes the element in the head of the queue.

Throws a `NoSuchElementException` if there is no element in the queue.

### Returns

the element in the head of the queue.

### Exceptions

<i>NoSuchElementException</i>	if there is no element in the queue.
-------------------------------	--------------------------------------

Implemented in `decaf::util::AbstractQueue< E>` (p. 178).

**6.656.3.2** `template<typename E> virtual bool decaf::util::Queue< E>::offer ( const`  
`E & value ) throw ( decaf::lang::exceptions::NullPointerException,`  
`decaf::lang::exceptions::IllegalArgumentException )` [pure  
virtual]

Inserts the specified element into the queue provided that the condition allows such an operation.

The method is generally preferable to the `collection.add(E)`, since the latter might throw an exception if the operation fails.

### Parameters

<i>value</i>	the specified element to insert into the queue.
--------------	---

### Returns

true if the operation succeeds and false if it fails.

### Exceptions

<i>NullPointerException</i>	if the <b>Queue</b> (p. 3239) implementation does not allow Null values to be inserted into the <b>Queue</b> (p. 3239).
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this queue

Implemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3834), and **decaf::util::PriorityQueue< E >** (p. 3121).

Referenced by **decaf::util::AbstractQueue< E >::add()**.

**6.656.3.3** `template<typename E> virtual bool decaf::util::Queue< E >::peek ( E & result ) const [pure virtual]`

Gets but not removes the element in the head of the queue.

The result if successful is assigned to the result parameter.

### Parameters

<i>result</i>	Reference to an instance of the contained type to assigned the removed value to.
---------------	--

### Returns

true if the element at the head of the queue was removed and assigned to the result parameter.

Implemented in **decaf::util::PriorityQueue< E >** (p. 3122).

Referenced by **decaf::util::AbstractQueue< E >::element()**.

**6.656.3.4** `template<typename E> virtual bool decaf::util::Queue< E >::poll ( E & result ) [pure virtual]`

Gets and removes the element in the head of the queue.

If the operation succeeds the value of the element at the head of the **Queue** (p. 3239) is assigned to the result parameter and the method returns true. If the operation fails the method returns false and the value of the result parameter is undefined.

**Parameters**

<i>result</i>	Reference to an instance of the contained type to assigned the removed value to.
---------------	--

**Returns**

true if the element at the head of the queue was removed and assigned to the result parameter.

Implemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3835), and **decaf::util::PriorityQueue< E >** (p. 3122).

Referenced by **decaf::util::AbstractQueue< E >::clear()**, and **decaf::util::AbstractQueue< E >::remove()**.

```
6.656.3.5  template<typename E> virtual E decaf::util::Queue< E >::remove ( )
           throw ( decaf::lang::exceptions::NoSuchElementException ) [pure
           virtual]
```

Gets and removes the element in the head of the queue.

Throws a **NoSuchElementException** if there is no element in the queue.

**Returns**

the element in the head of the queue.

**Exceptions**

<i>NoSuchElementException</i>	if there is no element in the queue.
-------------------------------	--------------------------------------

Implemented in **decaf::util::AbstractQueue< E >** (p. 179), and **decaf::util::PriorityQueue< E >** (p. 3123).

The documentation for this class was generated from the following file:

- **src/main/decaf/util/Queue.h**

**6.657 cms::QueueBrowser Class Reference**

This class implements in interface for browsing the messages in a **Queue** (p. 3238) without removing them.

```
#include <src/main/cms/QueueBrowser.h>
```

Inheritance diagram for **cms::QueueBrowser**:

## Public Member Functions

- virtual `~QueueBrowser()`
- virtual const `Queue * getQueue()` const =0 throw ( cms::CMSException )
- virtual std::string `getMessageSelector()` const =0 throw ( cms::CMSException )
- virtual `cms::MessageEnumeration * getEnumeration()`=0 throw ( cms::CMSException )

*Gets a pointer to an Enumeration object for browsing the Messages currently in the Queue (p. 3238) in the order that a client would receive them.*

### 6.657.1 Detailed Description

This class implements in interface for browsing the messages in a **Queue** (p. 3238) without removing them. To browse the contents of the **Queue** (p. 3238) the client calls the `getEnumeration` method to retrieve a new instance of a **Queue** (p. 3238) Enumerator. The client then calls the `hasMoreMessages` method of the Enumeration, if it returns true the client can safely call the `nextMessage` method of the Enumeration instance.

```
Enumeration* enumeration = queueBrowser->getEnumeration() (p. 3244);
while( enumeration->hasMoreMessages() ) { cms::Message (p. 2614)* message =
enumeration->nextMessage();
// ... Do something with the Message (p. 2614).
delete message; }
```

Since

1.1

### 6.657.2 Constructor & Destructor Documentation

6.657.2.1 virtual cms::QueueBrowser::~QueueBrowser ( ) [inline, virtual]

### 6.657.3 Member Function Documentation

6.657.3.1 virtual cms::MessageEnumeration\* cms::QueueBrowser::getEnumeration ( ) throw ( cms::CMSException ) [pure virtual]

Gets a pointer to an Enumeration object for browsing the Messages currently in the **Queue** (p. 3238) in the order that a client would receive them.

The pointer returned is owned by the browser and should not be deleted by the client application.

**Returns**

a pointer to a **Queue** (p. 3238) Enumeration, this Pointer is owned by the **Queue-Browser** (p. 3243) and should not be deleted by the client.

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	if an internal error occurs.
--	------------------------------

Implemented in **activemq::core::ActiveMQQueueBrowser** (p. 485).

**6.657.3.2** `virtual std::string cms::QueueBrowser::getMessageSelector ( ) const throw ( cms::CMSEException ) [pure virtual]`

**Returns**

the MessageSelector that is used on when this browser was created or empty string if no selector was present.

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	if an internal error occurs.
--	------------------------------

Implemented in **activemq::core::ActiveMQQueueBrowser** (p. 485).

**6.657.3.3** `virtual const Queue* cms::QueueBrowser::getQueue ( ) const throw ( cms::CMSEException ) [pure virtual]`

**Returns**

the **Queue** (p. 3238) that this browser is listening on.

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	if an internal error occurs.
--	------------------------------

Implemented in **activemq::core::ActiveMQQueueBrowser** (p. 485).

The documentation for this class was generated from the following file:

- src/main/cms/QueueBrowser.h

## 6.658 decaf::util::Random Class Reference

**Random** (p. 3245) Value Generator which is used to generate a stream of pseudorandom numbers.



```
#include <src/main/decaf/util/Random.h>
```

Inheritance diagram for decaf::util::Random:

## Public Member Functions

- **Random ()**  
*Construct a random generator with the current time of day in milliseconds as the initial state.*
- **Random (unsigned long long seed)**  
*Construct a random generator with the given `seed` as the initial state.*
- **bool nextBoolean ()**  
*Answers the next pseudo-random, uniformly distributed boolean value generated by this generator.*
- **double nextDouble ()**  
*Generates a normally distributed random double number between 0.0 inclusively and 1.0 exclusively.*
- **float nextFloat ()**  
*Generates a normally distributed random float number between 0.0 inclusively and 1.0 exclusively.*
- **double nextGaussian ()**  
*Pseudo-randomly generates (approximately) a normally distributed `double` value with mean 0.0 and a standard deviation value of 1.0 using the polar method of G.*
- **int nextInt ()**  
*Generates a uniformly distributed 32-bit `int` value from the this random number sequence.*
- **int nextInt (int n)**  
*Returns to the caller a new pseudo-random integer value which is uniformly distributed between 0 (inclusively) and the value of `n` (exclusively).*
- **long long nextLong ()**  
*Generates a uniformly distributed 64-bit `int` value from the this random number sequence.*
- **virtual void nextBytes (std::vector< unsigned char > &buf)**  
*Modifies the byte array by a random sequence of bytes generated by this random number generator.*
- **virtual void nextBytes (unsigned char \*buf, int size)**

*Modifies the byte array by a random sequence of bytes generated by this random number generator.*

- virtual void **setSeed** (unsigned long long seed)

*Modifies the seed using linear congruential formula presented in *The Art of Computer Programming, Volume 2, Section 3.2.1.**

## Protected Member Functions

- virtual int **next** (int bits)

*Answers a pseudo-random uniformly distributed `int` value of the number of bits specified by the argument `bits` as described by Donald E.*

### 6.658.1 Detailed Description

**Random** (p. 3245) Value Generator which is used to generate a stream of pseudorandom numbers. The algorithms implemented by class **Random** (p. 3245) use a protected utility method that on each invocation can supply up to 32 pseudorandomly generated bits.

#### Since

1.0

### 6.658.2 Constructor & Destructor Documentation

#### 6.658.2.1 `decaf::util::Random::Random ( )`

Construct a random generator with the current time of day in milliseconds as the initial state.

#### See also

**setSeed** (p. 3251)

#### 6.658.2.2 `decaf::util::Random::Random ( unsigned long long seed )`

Construct a random generator with the given `seed` as the initial state.

#### Parameters

<code>seed</code>	the seed that will determine the initial state of this random number generator
-------------------	--

#### See also

**setSeed** (p. 3251)

### 6.658.3 Member Function Documentation

#### 6.658.3.1 virtual int decaf::util::Random::next ( int *bits* ) [protected, virtual]

Answers a pseudo-random uniformly distributed `int` value of the number of bits specified by the argument `bits` as described by Donald E.

Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.2.1.

#### Returns

`int` a pseudo-random generated `int` number

#### Parameters

<i>bits</i>	number of bits of the returned value
-------------	--------------------------------------

#### See also

**nextBytes** (p. 3249)  
**nextDouble** (p. 3249)  
**nextFloat** (p. 3249)  
**nextInt()** (p. 3250)  
**nextInt(int)** (p. 3250)  
**nextGaussian** (p. 3249)  
**nextLong** (p. 3250)

Reimplemented in **decaf::security::SecureRandom** (p. 3427).

#### 6.658.3.2 bool decaf::util::Random::nextBoolean ( )

Answers the next pseudo-random, uniformly distributed boolean value generated by this generator.

#### Returns

boolean a pseudo-random, uniformly distributed boolean value

#### 6.658.3.3 virtual void decaf::util::Random::nextBytes ( unsigned char \* *buf*, int *size* ) [virtual]

Modifies the byte array by a random sequence of bytes generated by this random number generator.

#### Parameters

<i>buf</i>	non-null array to contain the new random bytes
------------	--

**See also****next** (p. 3247)**Exceptions**

<i>NullPointerException</i>	if buff is NULL
<i>IllegalArgumentException</i>	if size is negative

Reimplemented in **decaf::security::SecureRandom** (p. 3427).

**6.658.3.4** `virtual void decaf::util::Random::nextBytes ( std::vector< unsigned char > & buf )`  
 [virtual]

Modifies the byte array by a random sequence of bytes generated by this random number generator.

**Parameters**

<i>buf</i>	non-null array to contain the new random bytes
------------	--

**See also****next** (p. 3247)Reimplemented in **decaf::security::SecureRandom** (p. 3428).

**6.658.3.5** `double decaf::util::Random::nextDouble ( )`

Generates a normally distributed random double number between 0.0 inclusively and 1.0 exclusively.

**Returns**

double

**See also****nextFloat** (p. 3249)

**6.658.3.6** `float decaf::util::Random::nextFloat ( )`

Generates a normally distributed random float number between 0.0 inclusively and 1.0 exclusively.

**Returns**

float a random float number between 0.0 and 1.0

**See also**

**nextDouble** (p. 3249)

**6.658.3.7 double decaf::util::Random::nextGaussian ( )**

Pseudo-randomly generates (approximately) a normally distributed `double` value with mean 0.0 and a standard deviation value of 1.0 using the *polar method* of G.

E. P. Box, M. E. Muller, and G. Marsaglia, as described by Donald E. Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.4.1, subsection C, algorithm P

**Returns**

`double`

**See also**

**nextDouble** (p. 3249)

**6.658.3.8 int decaf::util::Random::nextInt ( )**

Generates a uniformly distributed 32-bit `int` value from the this random number sequence.

**Returns**

`int` uniformly distributed `int` value

**See also**

**next** (p. 3247)

**nextLong** (p. 3250)

**6.658.3.9 int decaf::util::Random::nextInt ( int n )**

Returns to the caller a new pseudo-random integer value which is uniformly distributed between 0 (inclusively) and the value of `n` (exclusively).

**Parameters**

<code>n</code>	The <code>int</code> value that defines the max value of the return.
----------------	--

**Returns**

the next pseudo random `int` value.

**Exceptions**

<i>IllegalArgumentException</i>	if <code>n</code> is less than or equal to zero.
---------------------------------	--

**6.658.3.10** `long long decaf::util::Random::nextLong ( )`

Generates a uniformly distributed 64-bit `int` value from the this random number sequence.

**Returns**

64-bit `int` random number

**See also**

**next** (p. 3247)  
**nextInt()** (p. 3250)  
**nextInt(int)** (p. 3250)

**6.658.3.11** `virtual void decaf::util::Random::setSeed ( unsigned long long seed )`  
[virtual]

Modifies the seed using linear congruential formula presented in *The Art of Computer Programming, Volume 2*, Section 3.2.1.

**Parameters**

<i>seed</i>	the seed that alters the state of the random number generator
-------------	---

**See also**

**next** (p. 3247)  
**Random()** (p. 3247)  
**#Random(long)**

Reimplemented in **decaf::security::SecureRandom** (p. 3429).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Random.h`

**6.659** `decaf::lang::Readable` Class Reference

A **Readable** (p. 3251) is a source of characters.

```
#include <src/main/decaf/lang/Readable.h>
```

Inheritance diagram for decaf::lang::Readable:

## Public Member Functions

- virtual **~Readable** ()
- virtual int **read** (**decaf::nio::CharBuffer** \*charBuffer)=0 throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException )

*Attempts to read characters into the specified character buffer.*

### 6.659.1 Detailed Description

A **Readable** (p. 3251) is a source of characters. Characters from a **Readable** (p. 3251) are made available to callers of the read method via a CharBuffer.

Since

1.0

### 6.659.2 Constructor & Destructor Documentation

6.659.2.1 virtual decaf::lang::Readable::~~Readable ( ) [inline, virtual]

### 6.659.3 Member Function Documentation

6.659.3.1 virtual int decaf::lang::Readable::read ( decaf::nio::CharBuffer \* *charBuffer* ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException ) [pure virtual]

Attempts to read characters into the specified character buffer.

The buffer is used as a repository of characters as-is: the only changes made are the results of a put operation. No flipping or rewinding of the buffer is performed.

#### Parameters

<i>charBuffer</i>	The Buffer to read Characters into.
-------------------	-------------------------------------

#### Returns

The number of char values added to the buffer, or -1 if this source of characters is at its end

**Exceptions**

<i>IOException</i>	- if an I/O error occurs
<i>NullPointerException</i>	- if buffer is NULL.
<i>ReadOnlyBufferException</i>	- if charBuffer is a read only buffer

Implemented in **decaf::io::Reader** (p. 3258).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Readable.h**

**6.660 activemq::transport::inactivity::ReadChecker Class Reference**

Runnable class that is used by the {.

```
#include <src/main/activemq/transport/inactivity/ReadChecker.h>
```

Inheritance diagram for activemq::transport::inactivity::ReadChecker:

**Public Member Functions**

- **ReadChecker** (**InactivityMonitor** \*parent)
- virtual **~ReadChecker** ()
- virtual void **run** ()

*Run method - called by the Thread class in the context of the thread.*

**6.660.1 Detailed Description**

Runnable class that is used by the {.

**See also**

**InactivityMonitor** (p. 2065)} class the check for timeouts related to **transport** (p. 99) reads.

**Since**

3.1



## 6.660.2 Constructor & Destructor Documentation

6.660.2.1 `activemq::transport::inactivity::ReadChecker::ReadChecker ( InactivityMonitor * parent )`

6.660.2.2 `virtual activemq::transport::inactivity::ReadChecker::~~ReadChecker ( )`  
[virtual]

## 6.660.3 Member Function Documentation

6.660.3.1 `virtual void activemq::transport::inactivity::ReadChecker::run ( )` [virtual]

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 3419).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/inactivity/ReadChecker.h`

## 6.661 decaf::io::Reader Class Reference

```
#include <src/main/decaf/io/Reader.h>
```

Inheritance diagram for decaf::io::Reader:

### Public Member Functions

- virtual `~Reader ()`
- virtual void **mark** (int readAheadLimit) throw ( decaf::io::IOException )  
*Marks the present position in the stream.*
- virtual bool **markSupported** () const  
*Tells whether this stream supports the **mark()** (p. 3256) operation.*
- virtual bool **ready** () const throw ( decaf::io::IOException )  
*Tells whether this stream is ready to be read.*
- virtual void **reset** () throw ( decaf::io::IOException )  
*Resets the stream.*
- virtual long long **skip** (long long count) throw ( decaf::io::IOException )  
*Skips characters.*
- virtual int **read** (std::vector< char > &buffer) throw ( decaf::io::IOException )

*Reads characters into an array.*

- virtual int **read** (char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )

*Reads characters into an array, the method will attempt to read as much data as the size of the array.*

- virtual int **read** (char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*Reads characters into a portion of an array.*

- virtual int **read** () throw ( decaf::io::IOException )

*Reads a single character.*

- virtual int **read** (decaf::nio::CharBuffer \*charBuffer) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException )

*Attempts to read characters into the specified character buffer.*

## Protected Member Functions

- **Reader** ()

- virtual int **doReadArrayBounded** (char \*buffer, int size, int offset, int length)=0 throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Override this method to customize the functionality of the method read( unsigned char\* buffer, int size, int offset, int length ).*

- virtual int **doReadVector** (std::vector< char > &buffer) throw ( decaf::io::IOException )

*Override this method to customize the functionality of the method read( std::vector<char>& buffer ) (p. 3259).*

- virtual int **doReadArray** (char \*buffer, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )

*Override this method to customize the functionality of the method read( char\* buffer, std::size\_t length ).*

- virtual int **doReadChar** () throw ( decaf::io::IOException )

*Override this method to customize the functionality of the method read() (p. 3257).*

- virtual int **doReadCharBuffer** (decaf::nio::CharBuffer \*charBuffer) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException )

*Override this method to customize the functionality of the method read( CharBuffer\* charBuffer ).*

### 6.661.1 Constructor & Destructor Documentation

6.661.1.1 `decaf::io::Reader::Reader ( )` [protected]

6.661.1.2 `virtual decaf::io::Reader::~~Reader ( )` [virtual]

### 6.661.2 Member Function Documentation

6.661.2.1 `virtual int decaf::io::Reader::doReadArray ( char * buffer, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )`  
[protected, virtual]

Override this method to customize the functionality of the method `read( char* buffer, std::size_t length )`.

6.661.2.2 `virtual int decaf::io::Reader::doReadArrayBounded ( char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`  
[protected, pure virtual]

Override this method to customize the functionality of the method `read( unsigned char* buffer, int size, int offset, int length )`.

All subclasses must override this method to provide the basic **Reader** (p. 3254) functionality.

Implemented in **decaf::io::InputStreamReader** (p. 2118).

6.661.2.3 `virtual int decaf::io::Reader::doReadChar ( ) throw ( decaf::io::IOException )`  
[protected, virtual]

Override this method to customize the functionality of the method `read()` (p. 3257).

6.661.2.4 `virtual int decaf::io::Reader::doReadCharBuffer ( decaf::nio::CharBuffer * charBuffer ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException )` [protected, virtual]

Override this method to customize the functionality of the method `read( CharBuffer* charBuffer )`.

6.661.2.5 `virtual int decaf::io::Reader::doReadVector ( std::vector< char > & buffer ) throw ( decaf::io::IOException )` [protected, virtual]

Override this method to customize the functionality of the method `read( std::vector<char>& buffer )` (p. 3259).

**6.661.2.6** `virtual void decaf::io::Reader::mark ( int readAheadLimit ) throw ( decaf::io::IOException )` [virtual]

Marks the present position in the stream.

Subsequent calls to **reset()** (p. 3260) will attempt to reposition the stream to this point. Not all character-input streams support the **mark()** (p. 3256) operation.

#### Parameters

<i>readAheadLimit</i>	Limit on the number of characters that may be read while still preserving the mark. After reading this many characters, attempting to reset the stream may fail.
-----------------------	--

#### Exceptions

<b>IOException</b> (p. 2210)	if an I/O error occurs, or the stream does not support mark.
---------------------------------	--

**6.661.2.7** `virtual bool decaf::io::Reader::markSupported ( ) const` [inline, virtual]

Tells whether this stream supports the **mark()** (p. 3256) operation.

The default implementation always returns false. Subclasses should override this method.

#### Returns

true if and only if this stream supports the mark operation.

**6.661.2.8** `virtual int decaf::io::Reader::read ( char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )` [virtual]

Reads characters into a portion of an array.

This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.

#### Parameters

<i>buffer</i>	The target char buffer.
<i>size</i>	The size in bytes of the target buffer.
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The maximum number of bytes to read.

#### Returns

The number of bytes read or -1 if the end of stream is reached.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	thrown if an I/O error occurs.
<i>NullPointerException</i>	if buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length is greater than the array size.

**6.661.2.9** virtual int decaf::io::Reader::read ( ) throw ( decaf::io::IOException )  
[virtual]

Reads a single character.

This method will block until a character is available, an I/O error occurs, or the end of the stream is reached.

Subclasses that intend to support efficient single-character input should override this method.

**Returns**

The character read, as an integer in the range 0 to 65535 (0x00-0xffff), or -1 if the end of the stream has been reached.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	thrown if an I/O error occurs.
--	--------------------------------

**6.661.2.10** virtual int decaf::io::Reader::read ( char \* *buffer*, int *size* ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException ) [virtual]

Reads characters into an array, the method will attempt to read as much data as the size of the array.

This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.

**Parameters**

<i>buffer</i>	The target char buffer.
<i>size</i>	The size in bytes of the target buffer.

**Returns**

The number of bytes read or -1 if the end of stream is reached.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	thrown if an I/O error occurs.
<b><i>NullPointerException</i></b>	if buffer is NULL.

**6.661.2.11** `virtual int decaf::io::Reader::read ( decaf::nio::CharBuffer * charBuffer ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Attempts to read characters into the specified character buffer.

The buffer is used as a repository of characters as-is: the only changes made are the results of a put operation. No flipping or rewinding of the buffer is performed.

**Parameters**

<i>charBuffer</i>	The Buffer to read Characters into.
-------------------	-------------------------------------

**Returns**

The number of char values added to the buffer, or -1 if this source of characters is at its end

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	- if an I/O error occurs
<b><i>NullPointerException</i></b>	- if buffer is NULL.
<b><i>ReadOnlyBufferException</i></b>	- if charBuffer is a read only buffer

Implements **decaf::lang::Readable** (p. 3252).

**6.661.2.12** `virtual int decaf::io::Reader::read ( std::vector< char > & buffer ) throw ( decaf::io::IOException ) [virtual]`

Reads characters into an array.

This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.

**Parameters**

<i>buffer</i>	The buffer to read characters into.
---------------	-------------------------------------

**Returns**

The number of characters read, or -1 if the end of the stream has been reached

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	thrown if an I/O error occurs.
--	--------------------------------

**6.661.2.13** `virtual bool decaf::io::Reader::ready ( ) const throw ( decaf::io::IOException )`  
[virtual]

Tells whether this stream is ready to be read.

**Returns**

True if the next **read()** (p. 3257) is guaranteed not to block for input, false otherwise. Note that returning false does not guarantee that the next read will block.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
--	-------------------------

Reimplemented in **decaf::io::InputStreamReader** (p. 2118).

**6.661.2.14** `virtual void decaf::io::Reader::reset ( ) throw ( decaf::io::IOException )`  
[virtual]

Resets the stream.

If the stream has been marked, then attempt to reposition it at the mark. If the stream has not been marked, then attempt to reset it in some way appropriate to the particular stream, for example by repositioning it to its starting point. Not all character-input streams support the **reset()** (p. 3260) operation, and some support **reset()** (p. 3260) without supporting **mark()** (p. 3256).

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
--	-------------------------

**6.661.2.15** `virtual long long decaf::io::Reader::skip ( long long count ) throw ( decaf::io::IOException )` [virtual]

Skips characters.

This method will block until some characters are available, an I/O error occurs, or the end of the stream is reached.

**Parameters**

<i>count</i>	The number of character to skip.
--------------	----------------------------------

**Returns**

the number of Character actually skipped.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
--	-------------------------

The documentation for this class was generated from the following file:

- src/main/decaf/io/**Reader.h**

**6.662 decaf::nio::ReadOnlyBufferException Class Reference**

```
#include <src/main/decaf/nio/ReadOnlyBufferException.h>
```

Inheritance diagram for decaf::nio::ReadOnlyBufferException:

**Public Member Functions**

- **ReadOnlyBufferException** () throw ()  
*Default Constructor.*
- **ReadOnlyBufferException** (const lang::Exception &ex) throw ()  
*Copy Constructor.*
- **ReadOnlyBufferException** (const **ReadOnlyBufferException** &ex) throw ()  
*Copy Constructor.*
- **ReadOnlyBufferException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **ReadOnlyBufferException** (const std::exception \*cause) throw ()  
*Constructor.*
- **ReadOnlyBufferException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **ReadOnlyBufferException** \* clone () const



*Clones this exception.*

- virtual `~ReadOnlyBufferException () throw ()`

### 6.662.1 Constructor & Destructor Documentation

**6.662.1.1** `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException ( ) throw ()`  
[inline]

Default Constructor.

**6.662.1.2** `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException ( const lang::Exception & ex ) throw ()` [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

**6.662.1.3** `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException ( const ReadOnlyBufferException & ex ) throw ()` [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

**6.662.1.4** `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()`  
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.662.1.5** `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.662.1.6** `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor.

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.662.1.7** `virtual decaf::nio::ReadOnlyBufferException::~~ReadOnlyBufferException ( ) throw () [inline, virtual]`

### 6.662.2 Member Function Documentation

**6.662.2.1** `virtual ReadOnlyBufferException* decaf::nio::ReadOnlyBufferException::clone ( )const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **decaf::lang::exceptions::UnsupportedOperationException** (p. 4030).

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/ReadOnlyBufferException.h`

## 6.663 decaf::util::concurrent::locks::ReadWriteLock Class Reference

A **ReadWriteLock** (p. 3263) maintains a pair of associated locks, one for read-only operations and one for writing.

```
#include <src/main/decaf/util/concurrent/locks/ReadWriteLock.h>
```

## Public Member Functions

- virtual **~ReadWriteLock** ()
- virtual **Lock & readLock** ()=0  
*Returns the lock used for reading.*
- virtual **Lock & writeLock** ()=0  
*Returns the lock used for writing.*

### 6.663.1 Detailed Description

A **ReadWriteLock** (p. 3263) maintains a pair of associated locks, one for read-only operations and one for writing. The read lock may be held simultaneously by multiple reader threads, so long as there are no writers. The write lock is exclusive.

All **ReadWriteLock** (p. 3263) implementations must guarantee that the memory synchronization effects of writeLock operations (as specified in the **Lock** (p. 2452) interface) also hold with respect to the associated readLock. That is, a thread successfully acquiring the read lock will see all updates made upon previous release of the write lock.

A read-write lock allows for a greater level of concurrency in accessing shared data than that permitted by a mutual exclusion lock. It exploits the fact that while only a single thread at a time (a writer thread) can modify the shared data, in many cases any number of threads can concurrently read the data (hence reader threads). In theory, the increase in concurrency permitted by the use of a read-write lock will lead to performance improvements over the use of a mutual exclusion lock. In practice this increase in concurrency will only be fully realized on a multi-processor, and then only if the access patterns for the shared data are suitable.

Whether or not a read-write lock will improve performance over the use of a mutual exclusion lock depends on the frequency that the data is read compared to being modified, the duration of the read and write operations, and the contention for the data - that is, the number of threads that will try to read or write the data at the same time. For example, a collection that is initially populated with data and thereafter infrequently modified, while being frequently searched (such as a directory of some kind) is an ideal candidate for the use of a read-write lock. However, if updates become frequent then the data spends most of its time being exclusively locked and there is little, if any increase in concurrency. Further, if the read operations are too short the overhead of the read-write lock implementation (which is inherently more complex than a mutual exclusion lock) can dominate the execution cost, particularly as many read-write lock implementations still serialize all threads through a small section of code. Ultimately, only profiling and measurement will establish whether the use of a read-write lock is suitable for your application.

Although the basic operation of a read-write lock is straight-forward, there are many policy decisions that an implementation must make, which may affect the effectiveness of the read-write lock in a given application. Examples of these policies include:

- \* Determining whether to grant the read lock or the write lock, when both readers and

writers are waiting, at the time that a writer releases the write lock. Writer preference is common, as writes are expected to be short and infrequent. Reader preference is less common as it can lead to lengthy delays for a write if the readers are frequent and long-lived as expected. Fair, or "in-order" implementations are also possible. \*

Determining whether readers that request the read lock while a reader is active and a writer is waiting, are granted the read lock. Preference to the reader can delay the writer indefinitely, while preference to the writer can reduce the potential for concurrency. \*

Determining whether the locks are reentrant: can a thread with the write lock reacquire it? Can it acquire a read lock while holding the write lock? Is the read lock itself reentrant? \*

Can the write lock be downgraded to a read lock without allowing an intervening writer? Can a read lock be upgraded to a write lock, in preference to other waiting readers or writers?

You should consider all of these things when evaluating the suitability of a given implementation for your application.

### Since

1.0

## 6.663.2 Constructor & Destructor Documentation

**6.663.2.1** `virtual decaf::util::concurrent::locks::ReadWriteLock::~~ReadWriteLock ( )`  
[inline, virtual]

## 6.663.3 Member Function Documentation

**6.663.3.1** `virtual Lock& decaf::util::concurrent::locks::ReadWriteLock::readLock ( )` [pure virtual]

Returns the lock used for reading.

### Returns

the lock used for reading.

**6.663.3.2** `virtual Lock& decaf::util::concurrent::locks::ReadWriteLock::writeLock ( )`  
[pure virtual]

Returns the lock used for writing.

### Returns

the lock used for writing.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/ReadWriteLock.h`

## 6.664 activemq::cmsutil::CmsTemplate::ReceiveExecutor Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for activemq::cmsutil::CmsTemplate::ReceiveExecutor:

### Public Member Functions

- **ReceiveExecutor** (**CmsTemplate** \*parent, **cms::Destination** \*destination, const std::string &selector, bool noLocal)
- virtual ~**ReceiveExecutor** ()
- virtual void **doInCms** (**cms::Session** \*session) throw (cms::CMSException)  
*Execute any number of operations against the supplied CMS session.*
- virtual **cms::Destination** \* **getDestination** (**cms::Session** \*session AMQCPP\_UNUSED) throw ( cms::CMSException )
- **cms::Message** \* **getMessage** ()

### Protected Member Functions

- **ReceiveExecutor** (const **ReceiveExecutor** &)
- **ReceiveExecutor** & **operator=** (const **ReceiveExecutor** &)

### Protected Attributes

- **cms::Destination** \* destination
- std::string selector
- bool noLocal
- **cms::Message** \* message
- **CmsTemplate** \* parent

### 6.664.1 Constructor & Destructor Documentation

- 6.664.1.1 `activemq::cmsutil::CmsTemplate::ReceiveExecutor::ReceiveExecutor ( const ReceiveExecutor & )` `[inline, protected]`
- 6.664.1.2 `activemq::cmsutil::CmsTemplate::ReceiveExecutor::ReceiveExecutor ( CmsTemplate * parent, cms::Destination * destination, const std::string & selector, bool noLocal )` `[inline]`
- 6.664.1.3 `virtual activemq::cmsutil::CmsTemplate::ReceiveExecutor::~ReceiveExecutor ( )` `[inline, virtual]`

### 6.664.2 Member Function Documentation

- 6.664.2.1 `virtual void activemq::cmsutil::CmsTemplate::ReceiveExecutor::doInCms ( cms::Session * session ) throw (cms::CMSEException)` `[virtual]`

Execute any number of operations against the supplied CMS session.

#### Parameters

<i>session</i>	the CMS Session
----------------	-----------------

#### Exceptions

<i>cms::CMSEException</i> (p. 1190)	if thrown by CMS API methods
--	------------------------------

Implements `activemq::cmsutil::SessionCallback` (p. 3476).

6.664.2.2 `virtual cms::Destination* activemq::cmsutil::CmsTemplate::ReceiveExecutor::getDestination ( cms::Session *session AMQCPP_UNUSED ) throw ( cms::CMSException )`  
`[inline, virtual]`

6.664.2.3 `cms::Message* activemq::cmsutil::CmsTemplate::ReceiveExecutor::getMessage ( )`  
`[inline]`

6.664.2.4 `ReceiveExecutor& activemq::cmsutil::CmsTemplate::ReceiveExecutor::operator= ( const ReceiveExecutor & )`  
`[inline, protected]`

### 6.664.3 Field Documentation

6.664.3.1 `cms::Destination* activemq::cmsutil::CmsTemplate::ReceiveExecutor::destination`  
`[protected]`

6.664.3.2 `cms::Message* activemq::cmsutil::CmsTemplate::ReceiveExecutor::message`  
`[protected]`

6.664.3.3 `bool activemq::cmsutil::CmsTemplate::ReceiveExecutor::noLocal`  
`[protected]`

6.664.3.4 `CmsTemplate* activemq::cmsutil::CmsTemplate::ReceiveExecutor::parent`  
`[protected]`

6.664.3.5 `std::string activemq::cmsutil::CmsTemplate::ReceiveExecutor::selector`  
`[protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

## 6.665 activemq::core::RedeliveryPolicy Class Reference

Interface for a **RedeliveryPolicy** (p. 3267) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.

```
#include <src/main/activemq/core/RedeliveryPolicy.h>
```

Inheritance diagram for `activemq::core::RedeliveryPolicy`:

### Public Member Functions

- `virtual ~RedeliveryPolicy ()`
- `virtual double getBackOffMultiplier () const =0`
- `virtual void setBackOffMultiplier (double value)=0`

*Sets the Back-Off Multiplier for Message Redelivery.*

- virtual short **getCollisionAvoidancePercent** () const =0
- virtual void **setCollisionAvoidancePercent** (short value)=0
- virtual long long **getInitialRedeliveryDelay** () const =0  
*Gets the initial time that redelivery of messages is delayed.*
- virtual void **setInitialRedeliveryDelay** (long long value)=0  
*Sets the initial time that redelivery will be delayed.*
- virtual int **getMaximumRedeliveries** () const =0  
*Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.*
- virtual void **setMaximumRedeliveries** (int maximumRedeliveries)=0  
*Sets the Maximum allowable redeliveries for a Message.*
- virtual long long **getRedeliveryDelay** (long long previousDelay)=0  
*Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.*
- virtual bool **isUseCollisionAvoidance** () const =0
- virtual void **setUseCollisionAvoidance** (bool value)=0
- virtual bool **isUseExponentialBackOff** () const =0
- virtual void **setUseExponentialBackOff** (bool value)=0
- virtual **RedeliveryPolicy** \* **clone** () const =0  
*Create a copy of this Policy and return it.*
- virtual void **configure** (const **decaf::util::Properties** &properties)  
*Checks the supplied properties object for properties matching the configurable settings of this class.*

## Static Public Attributes

- static const long long **NO\_MAXIMUM\_REDELIVERIES**

## Protected Member Functions

- **RedeliveryPolicy** ()

### 6.665.1 Detailed Description

Interface for a **RedeliveryPolicy** (p. 3267) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.



Since

3.2.0

## 6.665.2 Constructor & Destructor Documentation

6.665.2.1 `activemq::core::RedeliveryPolicy::RedeliveryPolicy ( )` `[protected]`

6.665.2.2 `virtual activemq::core::RedeliveryPolicy::~~RedeliveryPolicy ( )` `[virtual]`

## 6.665.3 Member Function Documentation

6.665.3.1 `virtual RedeliveryPolicy* activemq::core::RedeliveryPolicy::clone ( ) const`  
`[pure virtual]`

Create a copy of this Policy and return it.

### Returns

pointer to a new **RedeliveryPolicy** (p. 3267) that is a copy of this one.

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1731).

6.665.3.2 `virtual void activemq::core::RedeliveryPolicy::configure ( const`  
`decaf::util::Properties & properties )` `[virtual]`

Checks the supplied properties object for properties matching the configurable settings of this class.

The default implementation looks for properties named with the prefix `cms.RedeliveryPolicy.XXX` where `XXX` is the name of a property with a public setter method. For instance `cms.RedeliveryPolicy.useExponentialBackOff` will be used to set the value of the use exponential back off toggle.

Subclasses can override this method to add more configuration options or to exclude certain parameters from being set via the properties object.

### Parameters

<i>properties</i>	The Properties object used to configure this object.
-------------------	--

### Exceptions

<i>NumberFormatException</i>	if a property that is numeric cannot be converted
<i>IllegalArgumentException</i>	if a property can't be converted to the correct type.

**6.665.3.3** `virtual double activemq::core::RedeliveryPolicy::getBackOffMultiplier ( ) const`  
[pure virtual]

#### Returns

The value of the Back-Off Multiplier for Message Redelivery.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1732).

**6.665.3.4** `virtual short activemq::core::RedeliveryPolicy::getCollisionAvoidancePercent ( ) const`  
[pure virtual]

#### Returns

the currently set Collision Avoidance percentage.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1732).

**6.665.3.5** `virtual long long activemq::core::RedeliveryPolicy::getInitialRedeliveryDelay ( ) const`  
[pure virtual]

Gets the initial time that redelivery of messages is delayed.

#### Returns

the time in milliseconds that redelivery is delayed initially.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1732).

**6.665.3.6** `virtual int activemq::core::RedeliveryPolicy::getMaximumRedeliveries ( ) const`  
[pure virtual]

Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.

#### Returns

maximum allowed redeliveries for a message.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1732).

**6.665.3.7** `virtual long long activemq::core::RedeliveryPolicy::getRedeliveryDelay ( long long`  
*previousDelay* ) [pure virtual]

Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.

#### Parameters

<i>previousDelay</i>	The last delay that was used between message redeliveries.
----------------------	--

**Returns**

the new delay to use before attempting another redelivery.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1732).

**6.665.3.8** `virtual bool activemq::core::RedeliveryPolicy::isUseCollisionAvoidance ( ) const`  
[pure virtual]

**Returns**

whether or not collision avoidance is enabled for this Policy.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1733).

**6.665.3.9** `virtual bool activemq::core::RedeliveryPolicy::isUseExponentialBackOff ( ) const`  
[pure virtual]

**Returns**

whether or not the exponential back off option is enabled.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1733).

**6.665.3.10** `virtual void activemq::core::RedeliveryPolicy::setBackOffMultiplier ( double value )`  
[pure virtual]

Sets the Back-Off Multiplier for Message Redelivery.

**Parameters**

<i>value</i>	The new value for the back-off multiplier.
--------------	--

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1733).

**6.665.3.11** `virtual void activemq::core::RedeliveryPolicy::setCollisionAvoidancePercent ( short value )` [pure virtual]

**Parameters**

<i>value</i>	The collision avoidance percentage setting.
--------------	---

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1733).

**6.665.3.12** `virtual void activemq::core::RedeliveryPolicy::setInitialRedeliveryDelay ( long long value ) [pure virtual]`

Sets the initial time that redelivery will be delayed.

#### Parameters

<i>value</i>	Time in Milliseconds to wait before starting redelivery.
--------------	--

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1734).

**6.665.3.13** `virtual void activemq::core::RedeliveryPolicy::setMaximumRedeliveries ( int maximumRedeliveries ) [pure virtual]`

Sets the Maximum allowable redeliveries for a Message.

#### Parameters

<i>maximum-Redeliveries</i>	The maximum number of times that a message will be redelivered.
-----------------------------	---

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1734).

**6.665.3.14** `virtual void activemq::core::RedeliveryPolicy::setUseCollisionAvoidance ( bool value ) [pure virtual]`

#### Parameters

<i>value</i>	Enable or Disable collision avoidance for this Policy.
--------------	--

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1734).

**6.665.3.15** `virtual void activemq::core::RedeliveryPolicy::setUseExponentialBackOff ( bool value ) [pure virtual]`

#### Parameters

<i>value</i>	Enable or Disable the exponential back off multiplier option.
--------------	---

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1734).

## 6.665.4 Field Documentation

**6.665.4.1** `const long long activemq::core::RedeliveryPolicy::NO_MAXIMUM_REDELIVERIES [static]`

The documentation for this class was generated from the following file:

- src/main/activemq/core/RedeliveryPolicy.h

## 6.666 decaf::util::concurrent::locks::ReentrantLock Class Reference

A reentrant mutual exclusion **Lock** (p. 2452) with extended capabilities.

```
#include <src/main/decaf/util/concurrent/locks/ReentrantLock.h>
```

Inheritance diagram for decaf::util::concurrent::locks::ReentrantLock:

### Public Member Functions

- **ReentrantLock** ()
- virtual ~**ReentrantLock** ()
- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Acquires the lock.*
- virtual void **lockInterruptibly** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException )  
*Acquires the lock unless the current thread is interrupted.*
- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Acquires the lock only if it is not held by another thread at the time of invocation.*
- virtual bool **tryLock** (long long time, const **TimeUnit** &unit) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException )  
*Acquires the lock if it is not held by another thread within the given waiting time and the current thread has not been interrupted.*
- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Attempts to release this lock.*
- virtual **Condition** \* **newCondition** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::UnsupportedOperationException )  
*Returns a **Condition** (p. 1282) instance for use with this **Lock** (p. 2452) instance.*
- int **getHoldCount** () const  
*Queries the number of holds on this lock by the current thread.*
- bool **isHeldByCurrentThread** () const  
*Queries if this lock is held by the current thread.*
- bool **isLocked** () const

*Queries if this lock is held by any thread.*

- `bool isFair () const`

*Returns true if this lock has fairness set true.*

- `std::string toString () const`

*Returns a string identifying this lock, as well as its lock state.*

### 6.666.1 Detailed Description

A reentrant mutual exclusion **Lock** (p. 2452) with extended capabilities. A **ReentrantLock** (p. 3273) is owned by the thread last successfully locking, but not yet unlocking it. A thread invoking lock will return, successfully acquiring the lock, when the lock is not owned by another thread. The method will return immediately if the current thread already owns the lock. This can be checked using methods **isHeldByCurrentThread()** (p. 3275), and **getHoldCount()** (p. 3275).

The constructor for this class accepts an optional fairness parameter. When set true, under contention, locks favor granting access to the longest-waiting thread. Otherwise this lock does not guarantee any particular access order. Programs using fair locks accessed by many threads may display lower overall throughput (i.e., are slower; often much slower) than those using the default setting, but have smaller variances in times to obtain locks and guarantee lack of starvation. Note however, that fairness of locks does not guarantee fairness of thread scheduling. Thus, one of many threads using a fair lock may obtain it multiple times in succession while other active threads are not progressing and not currently holding the lock. Also note that the untimed tryLock method does not honor the fairness setting. It will succeed if the lock is available even if other threads are waiting.

It is recommended practice to always immediately follow a call to lock with a try block, most typically in a before/after construction such as:

```
class X { private:
```

```
    ReentrantLock (p. 3273) lock; // ...
```

```
public:
```

```
    void m() { lock.lock(); // block until condition holds
```

```
    try { // ... method body } finally { lock.unlock() } } }
```

In addition to implementing the **Lock** (p. 2452) interface, this class defines methods **isLocked** and **getLockQueueLength**, as well as some associated protected access methods that may be useful for instrumentation and monitoring.

**Since**

1.0

## 6.666.2 Constructor & Destructor Documentation

6.666.2.1 `decaf::util::concurrent::locks::ReentrantLock::ReentrantLock ( )`

6.666.2.2 `virtual decaf::util::concurrent::locks::ReentrantLock::~~ReentrantLock ( )`  
[virtual]

## 6.666.3 Member Function Documentation

6.666.3.1 `int decaf::util::concurrent::locks::ReentrantLock::getHoldCount ( ) const`

Queries the number of holds on this lock by the current thread.

A thread has a hold on a lock for each lock action that is not matched by an unlock action.

The hold count information is typically only used for testing and debugging purposes. For example, if a certain section of code should not be entered with the lock already held then we can assert that fact:

```
class X { private:
```

```
    ReentrantLock (p. 3273) lock; // ...
```

```
public:
```

```
void m() { assert( lock.getHoldCount() == 0 ); lock.lock(); try { // ... method body }  
catch(...) { lock.unlock(); } } }
```

### Returns

the number of holds on this lock by the current thread, or zero if this lock is not held by the current thread

6.666.3.2 `bool decaf::util::concurrent::locks::ReentrantLock::isFair ( ) const`

Returns true if this lock has fairness set true.

### Returns

true if this lock has fairness set true

6.666.3.3 `bool decaf::util::concurrent::locks::ReentrantLock::isHeldByCurrentThread ( ) const`

Queries if this lock is held by the current thread.

This method is typically used for debugging and testing. For example, a method that should only be called while a lock is held can assert that this is the case:

```
class X { private: ReentrantLock (p. 3273) lock = new ReentrantLock() (p. 3275); //  
... }
```

```
public: void m() { assert( lock.isHeldByCurrentThread() ); // ... method body } }
```

It can also be used to ensure that a reentrant lock is used in a non-reentrant manner, for example:

```
class X { private: ReentrantLock (p. 3273) lock = new ReentrantLock() (p. 3275); //
...
public: void m() { assert !lock.isHeldByCurrentThread() (p. 3275); lock.lock(); try {
// ... method body } finally { lock.unlock(); } } }
```

### Returns

true if current thread holds this lock and false otherwise

#### 6.666.3.4 `bool decaf::util::concurrent::locks::ReentrantLock::isLocked ( ) const`

Queries if this lock is held by any thread.

This method is designed for use in monitoring of the system state, not for synchronization control.

### Returns

true if any thread holds this lock and false otherwise

#### 6.666.3.5 `virtual void decaf::util::concurrent::locks::ReentrantLock::lock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [virtual]`

Acquires the lock.

Acquires the lock if it is not held by another thread and returns immediately, setting the lock hold count to one.

If the current thread already holds the lock then the hold count is incremented by one and the method returns immediately.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until the lock has been acquired, at which time the lock hold count is set to one.

### Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
-------------------------	--

Implements `decaf::util::concurrent::locks::Lock` (p. 2454).



**6.666.3.6** `virtual void decaf::util::concurrent::locks::ReentrantLock::lockInterruptibly  
( ) throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::InterruptedException ) [virtual]`

Acquires the lock unless the current thread is interrupted.

Acquires the lock if it is not held by another thread and returns immediately, setting the lock hold count to one.

If the current thread already holds this lock then the hold count is incremented by one and the method returns immediately.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

\* The lock is acquired by the current thread; or \* Some other thread interrupts the current thread.

If the lock is acquired by the current thread then the lock hold count is set to one.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while acquiring the lock,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

In this implementation, as this method is an explicit interruption point, preference is given to responding to the interrupt over normal or reentrant acquisition of the lock.

### Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
<i>InterruptedException</i>	if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported).

Implements **decaf::util::concurrent::locks::Lock** (p. 2454).

**6.666.3.7** `virtual Condition* decaf::util::concurrent::locks::ReentrantLock::newCondition  
( ) throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Returns a **Condition** (p. 1282) instance for use with this **Lock** (p. 2452) instance.

The returned **Condition** (p. 1282) instance supports the same usages as do the **Mutex** (p. 2866) Class' methods (wait, notify, and notifyAll).

\* If this lock is not held when any of the **Condition** (p. 1282) waiting or signalling methods are called, then an IllegalMonitorStateException is thrown. \* When the condition waiting methods are called the lock is released and, before they return, the lock is reacquired and the lock hold count restored to what it was when the method was called. \* If a thread is interrupted while waiting then the wait will terminate, an InterruptedException will be thrown, and the thread's interrupted status will be cleared. \* Waiting threads are signaled in FIFO order. \* The ordering of lock reacquisition for

threads returning from waiting methods is the same as for threads initially acquiring the lock, which is in the default case not specified, but for fair locks favors those threads that have been waiting the longest.

### Exceptions

<i>RuntimeException</i>	if an error occurs while creating the <b>Condition</b> (p. 1282).
<i>UnsupportedOperationException</i>	if this <b>Lock</b> (p. 2452) implementation does not support conditions

Implements **decaf::util::concurrent::locks::Lock** (p. 2455).

#### 6.666.3.8 **std::string decaf::util::concurrent::locks::ReentrantLock::toString ( ) const**

Returns a string identifying this lock, as well as its lock state.

The state, in brackets, includes either the String "Unlocked" or the String "Locked by" followed by the name of the owning thread.

### Returns

a string identifying this lock, as well as its lock state

#### 6.666.3.9 **virtual bool decaf::util::concurrent::locks::ReentrantLock::tryLock ( long long *time*, const TimeUnit & *unit* ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException ) [virtual]**

Acquires the lock if it is not held by another thread within the given waiting time and the current thread has not been interrupted.

Acquires the lock if it is not held by another thread and returns immediately with the value true, setting the lock hold count to one. If this lock has been set to use a fair ordering policy then an available lock will not be acquired if any other threads are waiting for the lock. This is in contrast to the **tryLock()** (p. 3279) method. If you want a timed tryLock that does permit barging on a fair lock then combine the timed and un-timed forms together:

```
if (lock.tryLock() || lock.tryLock(timeout, unit) ) { ... }
```

If the current thread already holds this lock then the hold count is incremented by one and the method returns true.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

\* The lock is acquired by the current thread; or \* Some other thread interrupts the current thread; or \* The specified waiting time elapses

If the lock is acquired then the value true is returned and the lock hold count is set to one.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while acquiring the lock,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

In this implementation, as this method is an explicit interruption point, preference is given to responding to the interrupt over normal or reentrant acquisition of the lock, and over reporting the elapse of the waiting time.

### Parameters

<i>time</i>	the maximum time to wait for the lock
<i>unit</i>	the time unit of the time argument

### Returns

true if the lock was acquired and false if the waiting time elapsed before the lock was acquired

### Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
<i>InterruptedException</i>	if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported)

Implements **decaf::util::concurrent::locks::Lock** (p. 2456).

**6.666.3.10** `virtual bool decaf::util::concurrent::locks::ReentrantLock::tryLock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [virtual]`

Acquires the lock only if it is not held by another thread at the time of invocation.

Acquires the lock if it is not held by another thread and returns immediately with the value true, setting the lock hold count to one. Even when this lock has been set to use a fair ordering policy, a call to **tryLock()** (p. 3279) will immediately acquire the lock if it is available, whether or not other threads are currently waiting for the lock. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting for this lock, then use **tryLock(0, TimeUnit.SECONDS** (p. 3930)) which is almost equivalent (it also detects interruption).

If the current thread already holds this lock then the hold count is incremented by one and the method returns true.

If the lock is held by another thread then this method will return immediately with the value false.

### Returns

true if the lock was acquired and false otherwise

**Exceptions**

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
-------------------------	--

Implements **decaf::util::concurrent::locks::Lock** (p. 2457).

```
6.666.3.11 virtual void decaf::util::concurrent::locks::ReentrantLock::unlock
( ) throw ( decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException ) [virtual]
```

Attempts to release this lock.

If the current thread is the holder of this lock then the hold count is decremented. If the hold count is now zero then the lock is released. If the current thread is not the holder of this lock then `IllegalMonitorStateException` is thrown.

**Exceptions**

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
-------------------------	--

Implements **decaf::util::concurrent::locks::Lock** (p. 2457).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/ReentrantLock.h`

## 6.667 decaf::util::concurrent::RejectedExecutionException Class Reference

```
#include <src/main/decaf/util/concurrent/RejectedExecutionException.h>
```

Inheritance diagram for `decaf::util::concurrent::RejectedExecutionException`:

**Public Member Functions**

- **RejectedExecutionException** () throw ()  
*Default Constructor.*
- **RejectedExecutionException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **RejectedExecutionException** (const **RejectedExecutionException** &ex) throw ()  
*Copy Constructor.*

## 6.667 decaf::util::concurrent::RejectedExecutionException Class Reference 3285

- **RejectedExecutionException** (const std::exception \***cause**) throw ()  
*Constructor.*
- **RejectedExecutionException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **RejectedExecutionException** (const char \*file, const int lineNumber, const std::exception \***cause**, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **RejectedExecutionException** \* **clone** () const  
*Clones this exception.*
- virtual ~**RejectedExecutionException** () throw ()

### 6.667.1 Constructor & Destructor Documentation

6.667.1.1 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException ( ) throw () [inline]

Default Constructor.

6.667.1.2 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException ( const Exception & **ex** ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.667.1.3 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException ( const RejectedExecutionException & **ex** ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	- The Exception to copy in this new instance.
-----------	---

6.667.1.4 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException ( const std::exception \* **cause** ) throw () [inline]

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.667.1.5** `decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.
<i>msg</i>	- The message to report
<i>...</i>	- list of primitives that are formatted into the message

**6.667.1.6** `decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.
<i>cause</i>	- The exception that was the cause for this one to be thrown.
<i>msg</i>	- The message to report
<i>...</i>	- list of primitives that are formatted into the message

**6.667.1.7** `virtual decaf::util::concurrent::RejectedExecutionException::~~RejectedExecutionException ( ) throw () [inline, virtual]`

**6.667.2 Member Function Documentation**

**6.667.2.1** `virtual RejectedExecutionException* decaf::util::concurrent::RejectedExecutionException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception

as well as the message. All subclasses should override.

### Returns

A new instance this exception type with a copy the current state.

Reimplemented from **decaf::lang::Exception** (p. 1889).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**RejectedExecutionException.h**

## 6.668 decaf::util::concurrent::RejectedExecutionHandler Class Reference

A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. ??).

```
#include <src/main/decaf/util/concurrent/RejectedExecutionHandler.h>
```

### Public Member Functions

- virtual **~RejectedExecutionHandler** ()
- virtual void **rejectedExecution** (Runnable \*r, ThreadPoolExecutor \*executer)=0  
throw ( RejectedExecutionException )

*Method that may be invoked by a **ThreadPoolExecutor** (p. ??) when **execute** (p. ??) cannot accept a task.*

### 6.668.1 Detailed Description

A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. ??).

### Since

1.0

## 6.668.2 Constructor & Destructor Documentation

6.668.2.1 `virtual decaf::util::concurrent::RejectedExecutionHandler::~~RejectedExecutionHandler ( ) [inline, virtual]`

## 6.668.3 Member Function Documentation

6.668.3.1 `virtual void decaf::util::concurrent::RejectedExecutionHandler::rejectedExecution ( Runnable * r, ThreadPoolExecutor * executor ) throw ( RejectedExecutionException ) [pure virtual]`

Method that may be invoked by a **ThreadPoolExecutor** (p. ??) when **execute** (p. ??) cannot accept a task.

This may occur when no more threads or queue slots are available because their bounds would be exceeded, or upon shutdown of the **Executor** (p. 1926).

In the absence of other alternatives, the method may throw an **RejectedExecutionException** (p. 3280), which will be propagated to the caller of **execute** (p. ??).

### Parameters

<i>r</i>	The pointer to the runnable task requested to be executed.
<i>executor</i>	The pointer to the executor attempting to execute this task.

### Exceptions

<b><i>RejectedExecutionException</i></b> (p. 3280)	if there is no remedy.
---	------------------------

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/RejectedExecutionHandler.h`

## 6.669 activemq::commands::RemoveInfo Class Reference

```
#include <src/main/activemq/commands/RemoveInfo.h>
```

Inheritance diagram for `activemq::commands::RemoveInfo`:

### Public Member Functions

- **RemoveInfo** ()
- virtual **~RemoveInfo** ()
- virtual unsigned char **getDataStructureType** () const

*Get the unique identifier that this object and its own Marshaler share.*



- virtual **RemoveInfo** \* **cloneDataStructure** () const

*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*

- virtual void **copyDataStructure** (const **DataStructure** \*src)

*Copy the contents of the passed object into this object's members, overwriting any existing data.*

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** \*value) const

*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **DataStructure** > & **getObjectId** () const

- virtual **Pointer**< **DataStructure** > & **getObjectId** ()

- virtual void **setObjectId** (const **Pointer**< **DataStructure** > &objectId)

- virtual long long **getLastDeliveredSequenceId** () const

- virtual void **setLastDeliveredSequenceId** (long long lastDeliveredSequenceId)

- virtual bool **isRemoveInfo** () const

- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

## Static Public Attributes

- static const unsigned char **ID\_REMOVEINFO** = 12

## Protected Attributes

- **Pointer**< **DataStructure** > **objectId**
- long long **lastDeliveredSequenceId**

### 6.669.1 Constructor & Destructor Documentation

6.669.1.1 `activemq::commands::RemoveInfo::RemoveInfo ( )`

6.669.1.2 `virtual activemq::commands::RemoveInfo::~~RemoveInfo ( )` [virtual]

### 6.669.2 Member Function Documentation

6.669.2.1 `virtual RemoveInfo* activemq::commands::RemoveInfo::cloneDataStructure ( )`  
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1714).

6.669.2.2 `virtual void activemq::commands::RemoveInfo::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

6.669.2.3 `virtual bool activemq::commands::RemoveInfo::equals ( const DataStructure * value ) const` [virtual]

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

6.669.2.4 `virtual unsigned char activemq::commands::RemoveInfo::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataSetructure** (p. 1713) type copy.

Implements **activemq::commands::DataSetructure** (p. 1717).

6.669.2.5 `virtual long long activemq::commands::RemoveInfo::getLastDeliveredSequenceld ( ) const [virtual]`

6.669.2.6 `virtual const Pointer<DataSetructure>& activemq::commands::RemoveInfo::getObjectld ( ) const [virtual]`

6.669.2.7 `virtual Pointer<DataSetructure>& activemq::commands::RemoveInfo::getObjectld ( ) [virtual]`

6.669.2.8 `virtual bool activemq::commands::RemoveInfo::isRemoveInfo ( ) const [inline, virtual]`

#### Returns

an answer of true to the **isRemoveInfo()** (p. 3286) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 769).

6.669.2.9 `virtual void activemq::commands::RemoveInfo::setLastDeliveredSequenceld ( long long lastDeliveredSequenceld ) [virtual]`

6.669.2.10 `virtual void activemq::commands::RemoveInfo::setObjectld ( const Pointer<DataSetructure> & objectld ) [virtual]`

6.669.2.11 `virtual std::string activemq::commands::RemoveInfo::toString ( ) const [virtual]`

Returns a string containing the information for this **DataSetructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

**6.669.2.12** `virtual Pointer<Command> activemq::commands::RemoveInfo::visit  
( activemq::state::CommandVisitor * visitor ) throw (  
exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

### 6.669.3 Field Documentation

**6.669.3.1** `const unsigned char activemq::commands::RemoveInfo::ID_-  
REMOVEINFO = 12 [static]`

**6.669.3.2** `long long activemq::commands::RemoveInfo::lastDeliveredSequenceId  
[protected]`

**6.669.3.3** `Pointer<DataStructure> activemq::commands::RemoveInfo::objectId  
[protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/RemoveInfo.h`

## 6.670 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3288).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfoMarsh
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller`:

#### Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const

*Creates a new instance of this marshalable type.*

- virtual unsigned char **getDataStructureType** () const

*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshall an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.670.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3288). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.670.2 Constructor & Destructor Documentation

6.670.2.1 `activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::RemoveInfoMarshaller ( ) [inline]`

6.670.2.2 `virtual activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::~~RemoveInfoMarshaller ( ) [inline, virtual]`

## 6.670.3 Member Function Documentation

6.670.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.670.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.670.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 808).

**6.670.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 809).

**6.670.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 810).

```
6.670.3.6  virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 811).

```
6.670.3.7  virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 813).

The documentation for this class was generated from the following file:



- src/main/activemq/wireformat/openwire/marshal/v2/**RemoveInfoMarshaller.h**

## 6.671 activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3292).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/RemoveInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller:

### Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.671.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3292). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.671.2 Constructor & Destructor Documentation

6.671.2.1 **activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::RemoveInfoMarshaller**  
( ) [inline]

6.671.2.2 **virtual activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::~~RemoveInfoMarshaller**  
( ) [inline, virtual]

### 6.671.3 Member Function Documentation

6.671.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.671.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.671.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 801).

**6.671.3.4** virtual void activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::looseUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 802).

**6.671.3.5** virtual int activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::tightMarshal1 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 803).

```
6.671.3.6  virtual void activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 804).

```
6.671.3.7  virtual void activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 806).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**RemoveInfoMarshaller.h**

## 6.672 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3296).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller**:

## Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.672.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3296). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.672.2 Constructor & Destructor Documentation

6.672.2.1 **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::RemoveInfoMarshaller**  
 ( ) [inline]

6.672.2.2 **virtual activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::~~RemoveInfoMarshaller**  
 ( ) [inline, virtual]

### 6.672.3 Member Function Documentation

6.672.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::createObject** ( )  
 const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.672.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.672.3.3** virtual void **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::looseMarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
\* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**  
(p. 772).

**6.672.3.4** virtual void **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::looseUnmarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
\* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**  
(p. 774).

```

6.672.3.5 virtual int activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 775).

```

6.672.3.6 virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 776).



6.672.3.7 virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**RemoveInfoMarshaller.h**

## 6.673 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3300).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller:

#### Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.673.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3300). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.673.2 Constructor & Destructor Documentation

6.673.2.1 **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::RemoveInfoMarshaller**  
( ) [inline]

6.673.2.2 **virtual activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::~~RemoveInfoMarshaller**  
( ) [inline, virtual]

### 6.673.3 Member Function Documentation

6.673.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.673.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

## Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.673.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 787).

**6.673.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 788).

```
6.673.3.5  virtual int activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 789).

```
6.673.3.6  virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

## 6.674 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller

### Class Reference

3309

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 790).

**6.673.3.7** virtual void **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 792).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**RemoveInfoMarshaller.h**

## 6.674 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3305).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/RemoveInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller**:

## Public Member Functions

- **RemoveInfoMarshaller ()**
- virtual **~RemoveInfoMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn)** throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.674.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3305). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.674.2 Constructor & Destructor Documentation

6.674.2.1 `activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::RemoveInfoMarshaller ( ) [inline]`

6.674.2.2 `virtual activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::~~RemoveInfoMarshaller ( ) [inline, virtual]`

## 6.674.3 Member Function Documentation

6.674.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.674.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.674.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 794).

**6.674.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 795).

**6.674.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 796).

**6.674.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 797).

**6.674.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/RemoveInfoMarshaller.h`

## 6.675 `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3309).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarsh
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller`:

### Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.675.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3309). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.675.2 Constructor & Destructor Documentation

**6.675.2.1** `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::RemoveInfoMarshaller ( ) [inline]`

**6.675.2.2** `virtual activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::~~RemoveInfoMarshaller ( ) [inline, virtual]`

### 6.675.3 Member Function Documentation

**6.675.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.675.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.675.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 780).

**6.675.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 781).

**6.675.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 782).

**6.675.3.6** virtual void activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::tightMarshal2  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure  
 \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
 utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
 [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 783).

**6.675.3.7** virtual void activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::tightUnmarshal  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream  
 \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**RemoveInfoMarshaller.h**

## 6.676 **activemq::commands::RemoveSubscriptionInfo** Class Reference

```
#include <src/main/activemq/commands/RemoveSubscriptionInfo.h>
```

Inheritance diagram for **activemq::commands::RemoveSubscriptionInfo**:

**Public Member Functions**

- **RemoveSubscriptionInfo** ()
- virtual **~RemoveSubscriptionInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaller share.*
- virtual **RemoveSubscriptionInfo** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const std::string & **getSubscriptionName** () const

- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual bool **isRemoveSubscriptionInfo** () const
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_REMOVESUBSCRIPTIONINFO** = 9

### Protected Attributes

- **Pointer< ConnectionId > connectionId**
- std::string **subscriptionName**
- std::string **clientId**

## 6.676.1 Constructor & Destructor Documentation

6.676.1.1 **activemq::commands::RemoveSubscriptionInfo::RemoveSubscriptionInfo** ( )

6.676.1.2 **virtual activemq::commands::RemoveSubscriptionInfo::~~RemoveSubscriptionInfo** ( ) [virtual]

## 6.676.2 Member Function Documentation

6.676.2.1 **virtual RemoveSubscriptionInfo\* activemq::commands::RemoveSubscriptionInfo::cloneDataStructure** ( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

**6.676.2.2** `virtual void activemq::commands::RemoveSubscriptionInfo::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

**6.676.2.3** `virtual bool activemq::commands::RemoveSubscriptionInfo::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

**6.676.2.4** `virtual const std::string& activemq::commands::RemoveSubscriptionInfo::getClientId ( ) const [virtual]`

**6.676.2.5** `virtual std::string& activemq::commands::RemoveSubscriptionInfo::getClientId ( ) [virtual]`

**6.676.2.6** `virtual const Pointer<ConnectionId>& activemq::commands::RemoveSubscriptionInfo::getConnectionId ( ) const [virtual]`

**6.676.2.7** `virtual Pointer<ConnectionId>& activemq::commands::RemoveSubscriptionInfo::getConnectionId ( ) [virtual]`

**6.676.2.8** `virtual unsigned char activemq::commands::RemoveSubscriptionInfo::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1713) type copy.

Implements `activemq::commands::DataStructure` (p. 1717).



6.676.2.9 virtual const std::string& activemq::commands::RemoveSubscriptionInfo::getSubscriptionName ( ) const [virtual]

6.676.2.10 virtual std::string& activemq::commands::RemoveSubscriptionInfo::getSubscriptionName ( ) [virtual]

6.676.2.11 virtual bool activemq::commands::RemoveSubscriptionInfo::isRemoveSubscriptionInfo ( ) const [inline, virtual]

#### Returns

an answer of true to the **isRemoveSubscriptionInfo()** (p. 3316) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 769).

6.676.2.12 virtual void activemq::commands::RemoveSubscriptionInfo::setClientId ( const std::string & *clientId* ) [virtual]

6.676.2.13 virtual void activemq::commands::RemoveSubscriptionInfo::setConnectionId ( const Pointer< ConnectionId > & *connectionId* ) [virtual]

6.676.2.14 virtual void activemq::commands::RemoveSubscriptionInfo::setSubscriptionName ( const std::string & *subscriptionName* ) [virtual]

6.676.2.15 virtual std::string activemq::commands::RemoveSubscriptionInfo::toString ( ) const [virtual]

Returns a string containing the information for this **DataSet** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

6.676.2.16 virtual Pointer<Command> activemq::commands::RemoveSubscriptionInfo::visit ( activemq::state::CommandVisitor \* *visitor* ) throw ( exceptions::ActiveMQException ) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

### 6.676.3 Field Documentation

- 6.676.3.1 **std::string activemq::commands::RemoveSubscriptionInfo::clientId**  
[protected]
- 6.676.3.2 **Pointer<ConnectionId> activemq::commands::RemoveSubscriptionInfo::connectionId**  
[protected]
- 6.676.3.3 **const unsigned char activemq::commands::RemoveSubscriptionInfo::ID\_-REMOVESUBSCRIPTIONINFO = 9** [static]
- 6.676.3.4 **std::string activemq::commands::RemoveSubscriptionInfo::subscriptionName**  
[protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**RemoveSubscriptionInfo.h**

## 6.677 activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3318).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller:

### Public Member Functions

- **RemoveSubscriptionInfoMarshaller ()**
- virtual **~RemoveSubscriptionInfoMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)**  
throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.677.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3318).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.677.2 Constructor & Destructor Documentation

**6.677.2.1** **ativemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller**  
 ( ) [inline]

**6.677.2.2** **virtual ativemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::~~RemoveSubscriptionInfoMarshaller**  
 ( ) [inline, virtual]

### 6.677.3 Member Function Documentation

**6.677.3.1** **virtual commands::DataStructure\* ativemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **ativemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.677.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaller.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.677.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 787).

**6.677.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

6.677 ac-

activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller

Class Reference

3325

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**  
(p. 788).

```
6.677.3.5  virtual int activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**  
(p. 789).

```
6.677.3.6  virtual void activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**  
(p. 790).

```
6.677.3.7 virtual void activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 792).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**RemoveSubscriptionInfoMarshaller.h**

## 6.678 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3322).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/RemoveSubscriptionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller**:

#### Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*

## 6.678 ac-

**activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller**

### Class Reference

3327

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

## 6.678.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3322).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.678.2 Constructor & Destructor Documentation

6.678.2.1 **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller**  
( ) [inline]

6.678.2.2 **virtual activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::~~RemoveSubscriptionInfoMarshaller**  
( ) [inline, virtual]

## 6.678.3 Member Function Documentation

6.678.3.1 **virtual commands::DataStructure\* ac-**  
**activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.678.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.678.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 772).

**6.678.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---



## 6.678 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller Class Reference 3329

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 774).

**6.678.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 775).

**6.678.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 776).

**6.678.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/RemoveSubscriptionInfoMarshaller.h`

## 6.679 **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3326).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/RemoveSubscriptionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller**:

- **RemoveSubscriptionInfoMarshaller ()**
- virtual **~RemoveSubscriptionInfoMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.679.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3326).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.679.2 Constructor & Destructor Documentation

6.679.2.1 `activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller ( ) [inline]`

6.679.2.2 `virtual activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::~~RemoveSubscriptionInfoMarshaller ( ) [inline, virtual]`

## 6.679.3 Member Function Documentation

6.679.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.679.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.679.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## 6.679 ac- tivismq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller

### Class Reference 3333

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 808).

**6.679.3.4** `virtual void activismq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 809).

**6.679.3.5** `virtual int activismq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 810).

```
6.679.3.6 virtual void activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 811).

```
6.679.3.7 virtual void activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 813).

The documentation for this class was generated from the following file:

6.680 ac-

**activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller**

**Class Reference**

3335

- [src/main/activemq/wireformat/openwire/marshal/v2/RemoveSubscriptionInfoMarshaller.h](#)

## 6.680 **activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3331).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/RemoveSubscriptionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller**:

### Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.680.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3331).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.680.2 Constructor & Destructor Documentation

6.680.2.1 **activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller**  
 ( ) [inline]

6.680.2.2 **virtual activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::~~RemoveSubscriptionInfoMarshaller**  
 ( ) [inline, virtual]

### 6.680.3 Member Function Documentation

6.680.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.680.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.680.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::looseMarshal**  
 ( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.



**6.680 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller**  
**Class Reference** **3337**  
**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 801).

**6.680.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 802).

**6.680.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 803).

```
6.680.3.6 virtual void activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 804).

```
6.680.3.7 virtual void activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**6.681 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller**  
**Class Reference** **3339**  
**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 806).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**RemoveSubscriptionInfoMarshaller.h**

## 6.681 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3335).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/RemoveSubscriptionInfoMa
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller**:

### Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.681.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3335).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.681.2 Constructor & Destructor Documentation

- 6.681.2.1 **activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller**  
 ( ) [inline]
- 6.681.2.2 **virtual activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::~~RemoveSubscriptionInfoMarshaller**  
 ( ) [inline, virtual]

### 6.681.3 Member Function Documentation

- 6.681.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

- 6.681.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

6.681 ac-

activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller

Class Reference

3341

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

```
6.681.3.3 virtual void activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 794).

```
6.681.3.4 virtual void activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 795).

```

6.681.3.5  virtual int activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 796).

```

6.681.3.6  virtual void activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 797).

6.682 ac-

activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller

Class Reference

3343

6.681.3.7 virtual void activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::tightUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream  
\* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**RemoveSubscriptionInfoMarshaller.h**

## 6.682 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3339).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/RemoveSubscriptionInfoMa
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller**:

#### Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.682.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3339).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.682.2 Constructor & Destructor Documentation

6.682.2.1 **activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller**  
( ) [inline]

6.682.2.2 **virtual activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::~~RemoveSubscriptionInfoMarshaller**  
( ) [inline, virtual]

### 6.682.3 Member Function Documentation

6.682.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.



**6.682 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller**  
**Class Reference** **3345**  
**Returns**

---

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.682.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.682.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 780).

**6.682.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 781).

```
6.682.3.5  virtual int activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 782).

```
6.682.3.6  virtual void activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 783).

**6.682.3.7** virtual void activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**RemoveSubscriptionInfoMarshaller.h**

**6.683 activemq::commands::ReplayCommand Class Reference**

```
#include <src/main/activemq/commands/ReplayCommand.h>
```

Inheritance diagram for activemq::commands::ReplayCommand:

**Public Member Functions**

- **ReplayCommand ()**

- virtual **~ReplayCommand** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ReplayCommand \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual int **getFirstNakNumber** () const
- virtual void **setFirstNakNumber** (int firstNakNumber)
- virtual int **getLastNakNumber** () const
- virtual void **setLastNakNumber** (int lastNakNumber)
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_REPLAYCOMMAND** = 65

### Protected Attributes

- int **firstNakNumber**
- int **lastNakNumber**

### 6.683.1 Constructor & Destructor Documentation

6.683.1.1 `activemq::commands::ReplayCommand::ReplayCommand ( )`

6.683.1.2 `virtual activemq::commands::ReplayCommand::~~ReplayCommand ( )`  
[virtual]

### 6.683.2 Member Function Documentation

6.683.2.1 `virtual ReplayCommand* activemq::commands::ReplayCommand::cloneDataStructure ( ) const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1714).

6.683.2.2 `virtual void activemq::commands::ReplayCommand::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<code>src</code>	- Source Object
------------------	-----------------

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

6.683.2.3 `virtual bool activemq::commands::ReplayCommand::equals ( const DataStructure * value ) const` [virtual]

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

**6.683.2.4** `virtual unsigned char activemq::commands::ReplayCommand::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataSet** (p. 1713) type copy.

Implements **activemq::commands::DataSet** (p. 1717).

**6.683.2.5** `virtual int activemq::commands::ReplayCommand::getFirstNakNumber ( ) const [virtual]`

**6.683.2.6** `virtual int activemq::commands::ReplayCommand::getLastNakNumber ( ) const [virtual]`

**6.683.2.7** `virtual void activemq::commands::ReplayCommand::setFirstNakNumber ( int firstNakNumber ) [virtual]`

**6.683.2.8** `virtual void activemq::commands::ReplayCommand::setLastNakNumber ( int lastNakNumber ) [virtual]`

**6.683.2.9** `virtual std::string activemq::commands::ReplayCommand::toString ( ) const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

**6.683.2.10** `virtual Pointer<Command> activemq::commands::ReplayCommand::visit ( activemq::state::CommandVisitor * visitor ) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

6.684

**activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller**

Class Reference

3351

### 6.683.3 Field Documentation

6.683.3.1 **int activemq::commands::ReplayCommand::firstNakNumber**  
[protected]

6.683.3.2 **const unsigned char activemq::commands::ReplayCommand::ID\_-REPLAYCOMMAND = 65** [static]

6.683.3.3 **int activemq::commands::ReplayCommand::lastNakNumber**  
[protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ReplayCommand.h**

## 6.684 **activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3347).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ReplayCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller**:

### Public Member Functions

- **ReplayCommandMarshaller ()**
- virtual **~ReplayCommandMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)**  
**throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** **throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.684.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3347).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.684.2 Constructor & Destructor Documentation

- 6.684.2.1 **activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::ReplayCommandMarshaller**  
 ( ) [inline]
- 6.684.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::~~ReplayCommandMarshaller**  
 ( ) [inline, virtual]

### 6.684.3 Member Function Documentation

- 6.684.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

- 6.684.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.



6.684

**activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller**

**Class Reference**

**3353**

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.684.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 780).

**6.684.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 781).

```

6.684.3.5  virtual int activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 782).

```

6.684.3.6  virtual void activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 783).

6.685

activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller

Class Reference

3355

6.684.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ReplayCommandMarshaller.h**

## 6.685 activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3351).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ReplayCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller**:

#### Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.685.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p.3351).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.685.2 Constructor & Destructor Documentation

6.685.2.1 **activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::ReplayCommandMarshaller**  
( ) [inline]

6.685.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::~~ReplayCommandMarshaller**  
( ) [inline, virtual]

### 6.685.3 Member Function Documentation

6.685.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

6.685

**activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller**

**Class Reference**

3357

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.685.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.685.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 787).

**6.685.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 788).

```
6.685.3.5  virtual int activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 789).

```
6.685.3.6  virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

6.686

**activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller**

**Class Reference**

**3359**

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 790).

**6.685.3.7** `virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 792).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ReplayCommandMarshaller.h**

## 6.686 **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3355).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ReplayCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller**:

## Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.686.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3355).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



6.686

activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller

Class Reference

3361

## 6.686.2 Constructor & Destructor Documentation

6.686.2.1 `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::ReplayCommandMarshaller  
( ) [inline]`

6.686.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::~~ReplayCommandMarshaller  
( ) [inline, virtual]`

## 6.686.3 Member Function Documentation

6.686.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::createObject  
( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.686.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::getDataStructureType  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.686.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::looseMarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 808).

**6.686.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 809).

**6.686.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

6.686

activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller

Class Reference

3363

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 810).

**6.686.3.6** virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::tightMarshal2 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** )  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 811).

**6.686.3.7** virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 813).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ReplayCommandMarshaller.h`

## 6.687 `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3360).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ReplayCommandMa
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller`:

### Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

6.687

**activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller**

**Class Reference**

**3365**

---

*Write a object instance to data output stream.*

### 6.687.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3360).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.687.2 Constructor & Destructor Documentation

6.687.2.1 **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::ReplayCommandMarshaller**  
( ) [inline]

6.687.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::~~ReplayCommandMarshaller**  
( ) [inline, virtual]

### 6.687.3 Member Function Documentation

6.687.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.687.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.687.3.3 **virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 772).

```
6.687.3.4  virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 774).

```
6.687.3.5  virtual int activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

6.687

**activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller**

**Class Reference**

**3367**

**Returns**

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 775).

**6.687.3.6** virtual void **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::tightMarshal2**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
\* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*,  
**utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** )  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 776).

**6.687.3.7** virtual void **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::tightUnmarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream**  
\* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ReplayCommandMarshaller.h**

## 6.688 **activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3364).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ReplayCommandMa
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller**:

**Public Member Functions**

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*



6.688

**activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller**

**Class Reference**

**3369**

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.688.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3364).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.688.2 Constructor & Destructor Documentation

6.688.2.1 **activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::ReplayCommandMarshaller**  
( ) [inline]

6.688.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::~~ReplayCommandMarshaller**  
( ) [inline, virtual]

### 6.688.3 Member Function Documentation

6.688.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.688.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.688.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 801).

**6.688.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 802).

6.688

activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller

Class Reference

3371

```
6.688.3.5 virtual int activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 803).

```
6.688.3.6 virtual void activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 804).

```
6.688.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 806).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ReplayCommandMarshaller.h**

## 6.689 activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3368).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ReplayCommandMa
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller**:

#### Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*

6.689

activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller

Class Reference

3373

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.689.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3368).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.689.2 Constructor & Destructor Documentation

6.689.2.1 **activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::ReplayCommandMarshaller**  
( ) [inline]

6.689.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::~~ReplayCommandMarshaller**  
( ) [inline, virtual]

### 6.689.3 Member Function Documentation

6.689.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.689.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.689.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 794).

**6.689.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

6.689

activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller

Class Reference

3375

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 795).

```
6.689.3.5  virtual int activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 796).

```
6.689.3.6  virtual void activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 797).

**6.689.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ReplayCommandMarshaller.h`

## 6.690 **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor** Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor**:



## Public Member Functions

- **ResolveProducerExecutor** (**ProducerCallback** \**action*, **CmsTemplate** \**parent*, const std::string &*destinationName*)
- virtual ~**ResolveProducerExecutor** ()
- virtual **cms::Destination** \* **getDestination** (**cms::Session** \**session*) throw ( cms::CMSExcption )

## Protected Member Functions

- **ResolveProducerExecutor** & **operator=** (const **ResolveProducerExecutor** &)

## 6.690.1 Constructor & Destructor Documentation

6.690.1.1 **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::ResolveProducerExecutor** ( **ProducerCallback** \* *action*, **CmsTemplate** \* *parent*, const std::string & *destinationName* ) [inline]

6.690.1.2 virtual **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::~~ResolveProducerExecutor** ( ) [inline, virtual]

## 6.690.2 Member Function Documentation

6.690.2.1 virtual **cms::Destination**\* **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::getDestination** ( **cms::Session** \* *session* ) throw ( **cms::CMSExcption** ) [virtual]

6.690.2.2 **ResolveProducerExecutor**& **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::operator=** ( const **ResolveProducerExecutor** & ) [inline, protected]

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

## 6.691 activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor**:

## Public Member Functions

- **ResolveReceiveExecutor** (**CmsTemplate** \***parent**, const std::string &**selector**, bool **noLocal**, const std::string &**destinationName**)
- virtual ~**ResolveReceiveExecutor** ()
- virtual **cms::Destination** \***getDestination** (**cms::Session** \***session**) throw ( cms::CMSException )

## Protected Member Functions

- **ResolveReceiveExecutor** & **operator=** (const **ResolveReceiveExecutor** &)

## 6.691.1 Constructor & Destructor Documentation

6.691.1.1 **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::ResolveReceiveExecutor** ( **CmsTemplate** \* *parent*, const std::string & *selector*, bool *noLocal*, const std::string & *destinationName* ) [inline]

6.691.1.2 virtual **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::~~ResolveReceiveExecutor** ( ) [inline, virtual]

## 6.691.2 Member Function Documentation

6.691.2.1 virtual **cms::Destination**\* **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::getDestination** ( **cms::Session** \* *session* ) throw ( cms::CMSException ) [virtual]

6.691.2.2 **ResolveReceiveExecutor**& **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::operator=** ( const **ResolveReceiveExecutor** & ) [inline, protected]

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/CmsTemplate.h

## 6.692 decaf::internal::util::Resource Class Reference

Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.

```
#include <src/main/decaf/internal/util/Resource.h>
```

Inheritance diagram for decaf::internal::util::Resource:

## Public Member Functions

- virtual ~**Resource** ()

### 6.692.1 Detailed Description

Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.

Since

1.0

### 6.692.2 Constructor & Destructor Documentation

6.692.2.1 virtual decaf::internal::util::Resource::~Resource ( ) [virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**Resource.h**

## 6.693 decaf::internal::util::ResourceLifecycleManager Class Reference

```
#include <src/main/decaf/internal/util/ResourceLifecycleManager.h>
```

### Public Member Functions

- **ResourceLifecycleManager** ()
- virtual **~ResourceLifecycleManager** ()
- virtual void **addResource** (**Resource** \*value)

### Protected Member Functions

- virtual void **destroyResources** ()

### 6.693.1 Detailed Description

Since

1.0

### 6.693.2 Constructor & Destructor Documentation

6.693.2.1 `decaf::internal::util::ResourceLifecycleManager::ResourceLifecycleManager ( )`

6.693.2.2 `virtual decaf::internal::util::ResourceLifecycleManager::~~ResourceLifecycleManager ( )` [virtual]

### 6.693.3 Member Function Documentation

6.693.3.1 `virtual void decaf::internal::util::ResourceLifecycleManager::addResource ( Resource * value )` [virtual]

6.693.3.2 `virtual void decaf::internal::util::ResourceLifecycleManager::destroyResources ( )` [protected, virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/ResourceLifecycleManager.h`

## 6.694 activemq::cmsutil::ResourceLifecycleManager Class Reference

Manages the lifecycle of a set of CMS resources.

```
#include <src/main/activemq/cmsutil/ResourceLifecycleManager.h>
```

### Public Member Functions

- **ResourceLifecycleManager ( )**
- **virtual ~ResourceLifecycleManager ( )**  
*Destructor - calls destroy*
- **void addConnection (cms::Connection \*connection) throw ( cms::CMSException )**  
*Adds a connection so that its life will be managed by this object.*
- **void addSession (cms::Session \*session) throw ( cms::CMSException )**  
*Adds a session so that its life will be managed by this object.*
- **void addDestination (cms::Destination \*dest) throw ( cms::CMSException )**  
*Adds a destination so that its life will be managed by this object.*
- **void addMessageProducer (cms::MessageProducer \*producer) throw ( cms::CMSException )**  
*Adds a message producer so that its life will be managed by this object.*

- void **addMessageConsumer** (cms::MessageConsumer \*consumer) throw ( cms::CMSEException )  
*Adds a message consumer so that its life will be managed by this object.*
- void **destroy** () throw ( cms::CMSEException )  
*Closes and destroys the contained CMS resources.*
- void **releaseAll** ()  
*Releases all of the contained resources so that this object will no longer control their lifetimes.*

### Protected Member Functions

- **ResourceLifecycleManager** (const **ResourceLifecycleManager** &)
- **ResourceLifecycleManager** & **operator=** (const **ResourceLifecycleManager** &)

#### 6.694.1 Detailed Description

Manages the lifecycle of a set of CMS resources. A call to `destroy` will close and destroy all of the contained resources in the appropriate manner.

#### 6.694.2 Constructor & Destructor Documentation

6.694.2.1 **activemq::cmsutil::ResourceLifecycleManager::ResourceLifecycleManager** ( const **ResourceLifecycleManager** & ) [`inline`, `protected`]

6.694.2.2 **activemq::cmsutil::ResourceLifecycleManager::ResourceLifecycleManager** ( )

6.694.2.3 **virtual activemq::cmsutil::ResourceLifecycleManager::~~ResourceLifecycleManager** ( ) [`virtual`]

Destructor - calls `destroy`

#### 6.694.3 Member Function Documentation

6.694.3.1 **void activemq::cmsutil::ResourceLifecycleManager::addConnection** ( cms::Connection \* *connection* ) throw ( cms::CMSEException )

Adds a connection so that its life will be managed by this object.

#### Parameters

<i>connection</i>	the object to be managed
-------------------	--------------------------

**6.694.3.2** void activemq::cmsutil::ResourceLifecycleManager::addDestination ( cms::Destination \* *dest* ) throw ( cms::CMSException )

Adds a destination so that its life will be managed by this object.

#### Parameters

<i>dest</i>	the object to be managed
-------------	--------------------------

**6.694.3.3** void activemq::cmsutil::ResourceLifecycleManager::addMessageConsumer ( cms::MessageConsumer \* *consumer* ) throw ( cms::CMSException )

Adds a message consumer so that its life will be managed by this object.

#### Parameters

<i>consumer</i>	the object to be managed
-----------------	--------------------------

**6.694.3.4** void activemq::cmsutil::ResourceLifecycleManager::addMessageProducer ( cms::MessageProducer \* *producer* ) throw ( cms::CMSException )

Adds a message producer so that its life will be managed by this object.

#### Parameters

<i>producer</i>	the object to be managed
-----------------	--------------------------

**6.694.3.5** void activemq::cmsutil::ResourceLifecycleManager::addSession ( cms::Session \* *session* ) throw ( cms::CMSException )

Adds a session so that its life will be managed by this object.

#### Parameters

<i>session</i>	the object to be managed
----------------	--------------------------

**6.694.3.6** void activemq::cmsutil::ResourceLifecycleManager::destroy ( ) throw ( cms::CMSException )

Closes and destroys the contained CMS resources.

#### Exceptions

<i>cms::CMSException</i> (p. 1190)	thrown if an error occurs.
---------------------------------------	----------------------------

6.694.3.7 **ResourceLifecycleManager& activemq::cmsutil::ResourceLifecycleManager::operator= ( const ResourceLifecycleManager & )** [inline, protected]

6.694.3.8 **void activemq::cmsutil::ResourceLifecycleManager::releaseAll ( )**

Releases all of the contained resources so that this object will no longer control their lifetimes.

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**ResourceLifecycleManager.h**

## 6.695 activemq::commands::Response Class Reference

```
#include <src/main/activemq/commands/Response.h>
```

Inheritance diagram for activemq::commands::Response:

### Public Member Functions

- **Response ( )**
- virtual **~Response ( )**
- virtual unsigned char **getDataStructureType ( )** const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **Response \* cloneDataStructure ( )** const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure (const DataStructure \*src)**  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString ( )** const  
*Returns a string containing the information for this DataStructure (p. 1713) such as its type and value of its elements.*
- virtual bool **equals (const DataStructure \*value)** const  
*Compares the DataStructure (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual int **getCorrelationId ( )** const
- virtual void **setCorrelationId (int correlationId)**
- virtual bool **isResponse ( )** const

- virtual **Pointer< Command > visit** (**activemq::state::CommandVisitor** \*visitor)  
throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_RESPONSE** = 30

### Protected Attributes

- int **correlationId**

## 6.695.1 Constructor & Destructor Documentation

6.695.1.1 **activemq::commands::Response::Response** ( )

6.695.1.2 **virtual activemq::commands::Response::~~Response** ( ) [virtual]

## 6.695.2 Member Function Documentation

6.695.2.1 **virtual Response\*** **activemq::commands::Response::cloneDataStructure** ( ) const  
[virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1573), **activemq::commands::DataResponse** (p. 1633), **activemq::commands::ExceptionResponse** (p. 1896), and **activemq::commands::IntegerResponse** (p. 2161).

6.695.2.2 **virtual void** **activemq::commands::Response::copyDataStructure** ( const  
**DataStructure \* src** ) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<i>src</i>	- Source Object
------------	-----------------



Reimplemented from **activemq::commands::BaseCommand** (p. 766).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1573), **activemq::commands::DataResponse** (p. 1633), **activemq::commands::ExceptionResponse** (p. 1896), and **activemq::commands::IntegerResponse** (p. 2161).

**6.695.2.3** `virtual bool activemq::commands::Response::equals ( const DataStructure *  
value ) const` [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1573), **activemq::commands::DataResponse** (p. 1633), **activemq::commands::ExceptionResponse** (p. 1896), and **activemq::commands::IntegerResponse** (p. 2161).

**6.695.2.4** `virtual int activemq::commands::Response::getCorrelationId ( ) const`  
[virtual]

**6.695.2.5** `virtual unsigned char activemq::commands::Response::getDataStructureType ( )  
const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1574), **activemq::commands::DataResponse** (p. 1634), **activemq::commands::ExceptionResponse** (p. 1897), and **activemq::commands::IntegerResponse** (p. 2162).

**6.695.2.6** `virtual bool activemq::commands::Response::isResponse ( ) const` [inline,  
virtual]

#### Returns

an answer of true to the **isResponse()** (p. 3381) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 769).

**6.695.2.7** `virtual void activemq::commands::Response::setCorrelationId ( int correlationId )`  
[virtual]

**6.695.2.8** `virtual std::string activemq::commands::Response::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1574), **activemq::commands::DataResponse** (p. 1634), **activemq::commands::ExceptionResponse** (p. 1897), and **activemq::commands::IntegerResponse** (p. 2162).

**6.695.2.9** `virtual Pointer<Command> activemq::commands::Response::visit`  
`( activemq::state::CommandVisitor * visitor ) throw (`  
`exceptions::ActiveMQException )` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

### 6.695.3 Field Documentation

**6.695.3.1** `int activemq::commands::Response::correlationId` [protected]

**6.695.3.2** `const unsigned char activemq::commands::Response::ID_RESPONSE = 30`  
[static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**Response.h**

## 6.696 activemq::transport::mock::ResponseBuilder Class Reference

Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

```
#include <src/main/activemq/transport/mock/ResponseBuilder.h>
```

Inheritance diagram for activemq::transport::mock::ResponseBuilder:

## Public Member Functions

- virtual **~ResponseBuilder** ()
- virtual **Pointer< Response > buildResponse** (const **Pointer< Command >** &command)=0  
*Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.*
- virtual void **buildIncomingCommands** (const **Pointer< Command >** &command, **decaf::util::StlQueue< Pointer< Command > >** &queue)=0  
*When called the **ResponseBuilder** (p. 3382) must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.*

### 6.696.1 Detailed Description

Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

### 6.696.2 Constructor & Destructor Documentation

6.696.2.1 virtual activemq::transport::mock::ResponseBuilder::~ResponseBuilder ( )  
 [inline, virtual]

### 6.696.3 Member Function Documentation

6.696.3.1 virtual void activemq::transport::mock::ResponseBuilder::buildIncomingCommands ( const **Pointer< Command >** & command, **decaf::util::StlQueue< Pointer< Command > >** & queue ) [pure virtual]

When called the **ResponseBuilder** (p. 3382) must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.

#### Parameters

<i>command</i>	- The Command being sent to the Broker.
<i>queue</i>	- Queue of Command sent back from the broker.

Implemented in **activemq::wireformat::openwire::OpenWireResponseBuilder** (p. 2990).

**6.696.3.2** virtual **Pointer<Response>** **activemq::transport::mock::ResponseBuilder::buildResponse** (  
**const Pointer< Command > &command** ) [pure virtual]

Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.

#### Parameters

<i>command</i>	- The command to build a response for
----------------	---------------------------------------

#### Returns

A Response object pointer, or NULL if no response.

Implemented in **activemq::wireformat::openwire::OpenWireResponseBuilder** (p. 2991).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/mock/**ResponseBuilder.h**

## 6.697 **activemq::transport::correlator::ResponseCorrelator** Class Reference

This type of transport filter is responsible for correlating asynchronous responses with requests.

```
#include <src/main/activemq/transport/correlator/ResponseCorrelator.h>
```

Inheritance diagram for **activemq::transport::correlator::ResponseCorrelator**:

#### Public Member Functions

- **ResponseCorrelator** (const **Pointer< Transport > &next**)

*Constructor.*

- virtual **~ResponseCorrelator** ()
- virtual void **oneway** (const **Pointer< Command > &command**) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Sends a one-way command.*

- virtual **Pointer< Response > request** (const **Pointer< Command > &command**) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Sends the given request to the server and waits for the response.*

## 6.697 activemq::transport::correlator::ResponseCorrelator Class Reference 3389

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends the given request to the server and waits for the response.*
- virtual void **onCommand** (const **Pointer**< **Command** > &command)  
*This is called in the context of the nested transport's reading thread.*
- virtual void **start** () throw ( decaf::io::IOException )  
*Starts this transport object and creates the thread for polling on the input stream for commands.*
- virtual void **close** () throw ( decaf::io::IOException )  
*Stops the polling thread and closes the streams.*
- virtual void **onTransportException** (**Transport** \*source, const **decaf::lang::Exception** &ex)  
*Event handler for an exception from a command transport.*

### 6.697.1 Detailed Description

This type of transport filter is responsible for correlating asynchronous responses with requests. Non-response messages are simply sent directly to the CommandListener. It owns the transport that it

### 6.697.2 Constructor & Destructor Documentation

#### 6.697.2.1 activemq::transport::correlator::ResponseCorrelator::ResponseCorrelator ( const **Pointer**< **Transport** > & next )

Constructor.

#### Parameters

<i>next</i>	the next transport in the chain
-------------	---------------------------------

**6.697.2.2** `virtual activemq::transport::correlator::ResponseCorrelator::~~ResponseCorrelator ( ) [virtual]`

### 6.697.3 Member Function Documentation

**6.697.3.1** `virtual void activemq::transport::correlator::ResponseCorrelator::close ( ) throw ( decaf::io::IOException ) [virtual]`

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

#### Exceptions

<i>IOException</i>	if errors occur.
--------------------	------------------

Reimplemented from `activemq::transport::TransportFilter` (p. 4007).

**6.697.3.2** `virtual void activemq::transport::correlator::ResponseCorrelator::onCommand ( const Pointer< Command > & command ) [virtual]`

This is called in the context of the nested transport's reading thread.

In the case of a response object, updates the request map and notifies those waiting on the response. Non-response messages are just delegated to the command listener.

#### Parameters

<i>command</i>	the received from the nested transport.
----------------	---

Reimplemented from `activemq::transport::TransportFilter` (p. 4009).

**6.697.3.3** `virtual void activemq::transport::correlator::ResponseCorrelator::oneway ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

#### Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

#### Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

## 6.697 activemq::transport::correlator::ResponseCorrelator Class Reference 3391

Reimplemented from **activemq::transport::TransportFilter** (p. 4010).

**6.697.3.4** `virtual void activemq::transport::correlator::ResponseCorrelator::onTransportException ( Transport * source, const decaf::lang::Exception & ex ) [virtual]`

Event handler for an exception from a command transport.

### Parameters

<i>source</i>	The source of the exception
<i>ex</i>	The exception.

**6.697.3.5** `virtual Pointer<Response> activemq::transport::correlator::ResponseCorrelator::request ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Sends the given request to the server and waits for the response.

### Parameters

<i>command</i>	The request to send.
----------------	----------------------

### Returns

the response from the server.

### Exceptions

<i>IOException</i>	if an error occurs with the request.
--------------------	--------------------------------------

Reimplemented from **activemq::transport::TransportFilter** (p. 4011).

**6.697.3.6** `virtual Pointer<Response> activemq::transport::correlator::ResponseCorrelator::request ( const Pointer< Command > & command, unsigned int timeout ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Sends the given request to the server and waits for the response.

### Parameters

<i>command</i>	The request to send.
<i>timeout</i>	The time to wait for a response.

**Returns**

the response from the server.

**Exceptions**

<i>IOException</i>	if an error occurs with the request.
--------------------	--------------------------------------

Reimplemented from **activemq::transport::TransportFilter** (p. 4011).

**6.697.3.7** `virtual void activemq::transport::correlator::ResponseCorrelator::start ( ) throw ( decaf::io::IOException )` [virtual]

Starts this transport object and creates the thread for polling on the input stream for commands.

If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

**Exceptions**

<i>IOException</i>	if an error occurs or if this transport has already been closed.
--------------------	--

Reimplemented from **activemq::transport::TransportFilter** (p. 4012).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/correlator/**ResponseCorrelator.h**

## 6.698 **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3388).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ResponseMarshal
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller**:

**Public Member Functions**

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const



*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.698.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p.3388). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.698.2 Constructor & Destructor Documentation

6.698.2.1 `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::ResponseMarshaller ( ) [inline]`

6.698.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::~~ResponseMarshaller ( ) [inline, virtual]`

## 6.698.3 Member Function Documentation

6.698.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1584), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1653), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1916), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2177).

6.698.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1585), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1653), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1916), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2177).

6.698.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

## 6.698 activemq::wireformat::openwire::marshal::v4::ResponseMarshaller Class Reference 3395

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 780).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1585), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1654), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1917), and **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 2177).

```
6.698.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::looseUnmarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
  * dataStructure, decaf::io::DataInputStream * dataIn ) throw (  
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 781).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1585), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1654), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1917), and **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 2178).

6.698.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 782).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1586), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1654), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1917), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2178).

6.698.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.699 `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` Class Reference 3397

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 783).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1586), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1655), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1918), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2179).

```
6.698.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 784).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1587), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1655), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1918), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2179).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h`

## 6.699 `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3393).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller`:

## Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**)  
**throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) **throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) **throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) **throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) **throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.699.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3393). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.699.2 Constructor & Destructor Documentation

6.699.2.1 `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::ResponseMarshaller  
( ) [inline]`

6.699.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::~~ResponseMarshaller  
( ) [inline, virtual]`

## 6.699.3 Member Function Documentation

6.699.3.1 `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v2::ResponseMarshaller::createObject ( )  
const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1576), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1644), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1903), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2168).

6.699.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::getDataStructureType  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1576), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1645), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1903), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2168).

6.699.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::looseMarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 808).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1576), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1645), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1904), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2169).

```
6.699.3.4  virtual void activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataInputStream * dataIn ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 809).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1577), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1645), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1904), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2169).



## 6.699 activemq::wireformat::openwire::marshal::v2::ResponseMarshaller Class Reference 3401

**6.699.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 810).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1577), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1646), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1905), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2169).

**6.699.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 811).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1578), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1646), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1905), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2170).

**6.699.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 813).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1578), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1647), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1906), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2170).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h`

## 6.700 `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3398).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ResponseMarshal
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller`:

## Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**)  
throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.700.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3398). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.700.2 Constructor & Destructor Documentation

6.700.2.1 `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::ResponseMarshaller ( ) [inline]`

6.700.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::~~ResponseMarshaller ( ) [inline, virtual]`

## 6.700.3 Member Function Documentation

6.700.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1593), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1636), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1912), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2185).

6.700.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1593), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1636), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1912), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2186).

6.700.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

## 6.700 activemq::wireformat::openwire::marshal::v5::ResponseMarshaller Class Reference 3405

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 794).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller** (p. 1594), **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller** (p. 1636), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1912), and **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller** (p. 2186).

```
6.700.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
 * dataStructure, decaf::io::DataInputStream * dataIn ) throw (
 decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 795).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller** (p. 1594), **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller** (p. 1637), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1913), and **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller** (p. 2186).

```

6.700.3.5  virtual int activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 796).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1594), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1637), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1913), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2187).

```

6.700.3.6  virtual void activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.701 activemq::wireformat::openwire::marshal::v3::ResponseMarshaller Class Reference 3407

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 797).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1595), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1638), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1914), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2187).

```
6.700.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
  * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 799).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1595), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1638), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1914), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2188).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h`

## 6.701 activemq::wireformat::openwire::marshal::v3::ResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3403).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller`:

## Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**)  
**throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) **throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) **throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) **throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) **throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.701.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3403). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.701.2 Constructor & Destructor Documentation

**6.701.2.1** `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::ResponseMarshaller  
( ) [inline]`

**6.701.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::~~ResponseMarshaller  
( ) [inline, virtual]`

## 6.701.3 Member Function Documentation

**6.701.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v3::ResponseMarshaller::createObject ( )  
const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1580), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1649), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1907), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2172).

**6.701.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::getDataStructureType  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1580), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1649), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1908), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2173).

**6.701.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::looseMarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 772).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1581), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1649), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1908), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2173).

```
6.701.3.4  virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataInputStream * dataIn ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 774).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1581), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1650), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1908), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2173).

## 6.701 activemq::wireformat::openwire::marshal::v3::ResponseMarshaller Class Reference 3411

**6.701.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 775).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1581), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1650), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1909), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2174).

**6.701.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 776).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1582), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1651), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1909), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2174).

**6.701.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 777).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1582), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1651), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1910), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2175).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h`

## 6.702 `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3408).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ResponseMarshal
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller`:

## Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**)  
throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.702.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3408). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.702.2 Constructor & Destructor Documentation

6.702.2.1 `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::ResponseMarshaller ( ) [inline]`

6.702.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::~~ResponseMarshaller ( ) [inline, virtual]`

## 6.702.3 Member Function Documentation

6.702.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1589), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1657), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1920), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2181).

6.702.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1589), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1658), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1921), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2181).

6.702.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

## 6.702 activemq::wireformat::openwire::marshal::v1::ResponseMarshaller Class Reference 3415

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 787).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1589), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1658), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1921), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 2182).

```
6.702.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 788).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1590), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1658), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1921), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 2182).

6.702.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 789).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1590), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1659), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1922), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2182).

6.702.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



## 6.703 `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` Class Reference 3417

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 790).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1591), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1659), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1922), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2183).

```
6.702.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
  * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 792).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1591), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1660), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1923), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2183).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h`

## 6.703 `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3413).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller`:

## Public Member Functions

- **ResponseMarshaller ()**
- virtual **~ResponseMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.703.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3413). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.703.2 Constructor & Destructor Documentation

**6.703.2.1** `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::ResponseMarshaller  
( ) [inline]`

**6.703.2.2** `virtual activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::~~ResponseMarshaller  
( ) [inline, virtual]`

## 6.703.3 Member Function Documentation

**6.703.3.1** `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v6::ResponseMarshaller::createObject ( )  
const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1597), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1640), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1899), and `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2164).

**6.703.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::getDataStructureType  
( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1598), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1640), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1899), and `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2164).

**6.703.3.3** `virtual void activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::looseMarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 801).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1598), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1641), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1899), and `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2164).

```
6.703.3.4 virtual void activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 802).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1598), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1641), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1900), and `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2165).

## 6.703 activemq::wireformat::openwire::marshal::v6::ResponseMarshaller Class Reference 3421

**6.703.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 803).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1599), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1642), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1900), and `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2165).

**6.703.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 804).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1599), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1642), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1901), and `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2166).

**6.703.3.7** `virtual void activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 806).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1600), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1643), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1901), and `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2166).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h`

## 6.704 decaf::lang::Runnable Class Reference

Interface for a runnable object - defines a task that can be run by a thread.

```
#include <src/main/decaf/lang/Runnable.h>
```

Inheritance diagram for `decaf::lang::Runnable`:

## Public Member Functions

- virtual **~Runnable** ()
- virtual void **run** ()=0

*Run method - called by the **Thread** (p. 3876) class in the context of the thread.*

### 6.704.1 Detailed Description

Interface for a runnable object - defines a task that can be run by a thread.

### 6.704.2 Constructor & Destructor Documentation

6.704.2.1 virtual decaf::lang::Runnable::~Runnable ( ) [inline, virtual]

### 6.704.3 Member Function Documentation

6.704.3.1 virtual void decaf::lang::Runnable::run ( ) [pure virtual]

Run method - called by the **Thread** (p. 3876) class in the context of the thread.

Implemented in **activemq::threads::CompositeTaskRunner** (p. 1258), **activemq::threads::DedicatedTaskRunner** (p. 1725), **activemq::transport::inactivity::ReadChecker** (p. 3253), **activemq::transport::inactivity::WriteChecker** (p. 4133), **activemq::transport::IOTransport** (p. 2219), **activemq::transport::mock::InternalCommandListener** (p. 2193), **decaf::lang::Thread** (p. 3884), and **decaf::util::concurrent::PooledThread** (p. 3057).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Runnable.h**

## 6.705 decaf::lang::Runtime Class Reference

```
#include <src/main/decaf/lang/Runtime.h>
```

Inheritance diagram for decaf::lang::Runtime:

## Public Member Functions

- virtual **~Runtime** ()

## Static Public Member Functions

- static **Runtime \* getRuntime** ()

*Gets the single instance of the Decaf **Runtime** (p. 3419) for this Process.*

- static void **initializeRuntime** (int argc, char \*\*argv)

*Initialize the Decaf Library passing it the args that were passed to the application at startup.*

- static void **initializeRuntime** ()

*Initialize the Decaf Library.*

- static void **shutdownRuntime** ()

*Shutdown the Decaf Library, this call should take places after all objects that were created from the Decaf library have been deallocated.*

## 6.705.1 Constructor & Destructor Documentation

6.705.1.1 virtual decaf::lang::Runtime::~~Runtime ( ) [inline, virtual]

## 6.705.2 Member Function Documentation

6.705.2.1 static Runtime\* decaf::lang::Runtime::getRuntime ( ) [static]

Gets the single instance of the Decaf **Runtime** (p. 3419) for this Process.

### Returns

pointer to the single Decaf **Runtime** (p. 3419) instance that exists for this process

6.705.2.2 static void decaf::lang::Runtime::initializeRuntime ( int argc, char \*\* argv )  
[static]

Initialize the Decaf Library passing it the args that were passed to the application at startup.

### Parameters

<i>argc</i>	- The number of args passed
<i>argv</i>	- Array of char* values passed to the Process on start.

### Exceptions

<i>runtime_error</i>	if the library is already initialized or an error occurs during initialization.
----------------------	---



**6.705.2.3 static void decaf::lang::Runtime::initializeRuntime ( ) [static]**

Initialize the Decaf Library.

**Exceptions**

<i>runtime_error</i>	if the library is already initialized or an error occurs during initialization.
----------------------	---

**6.705.2.4 static void decaf::lang::Runtime::shutdownRuntime ( ) [static]**

Shutdown the Decaf Library, this call should take places after all objects that were created from the Decaf library have been deallocated.

**Exceptions**

<i>runtime_error</i>	if the library has not already been initialized or an error occurs during shutdown.
----------------------	---

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Runtime.h**

**6.706 decaf::lang::exceptions::RuntimeException Class Reference**

```
#include <src/main/decaf/lang/exceptions/RuntimeException.h>
```

Inheritance diagram for decaf::lang::exceptions::RuntimeException:

**Public Member Functions**

- **RuntimeException** () throw ()  
*Default Constructor.*
- **RuntimeException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other ActiveMQException.*
- **RuntimeException** (const **RuntimeException** &ex) throw ()  
*Copy Constructor.*
- **RuntimeException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*

- **RuntimeException** (const std::exception \***cause**) throw ()  
*Constructor.*
- **RuntimeException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **RuntimeException** \* **clone** () const  
*Clones this exception.*
- virtual ~**RuntimeException** () throw ()

### 6.706.1 Constructor & Destructor Documentation

6.706.1.1 **decaf::lang::exceptions::RuntimeException::RuntimeException ( ) throw ()**  
[inline]

Default Constructor.

6.706.1.2 **decaf::lang::exceptions::RuntimeException::RuntimeException ( const Exception & ex ) throw ()** [inline]

Conversion Constructor from some other ActiveMQException.

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

6.706.1.3 **decaf::lang::exceptions::RuntimeException::RuntimeException ( const RuntimeException & ex ) throw ()** [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1886) whose data is to be copied into this one.
-----------	--

6.706.1.4 **decaf::lang::exceptions::RuntimeException::RuntimeException ( const char \* file, const int lineNumber, const std::exception \* cause, const char \* msg, ... ) throw ()**  
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.706.1.5** decaf::lang::exceptions::RuntimeException::RuntimeException ( const std::exception \* *cause* ) throw () [inline]

Constructor.

**Parameters**

<i>cause</i>	<b>Pointer</b> (p. 3032) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

**6.706.1.6** decaf::lang::exceptions::RuntimeException::RuntimeException ( const char \* *file*, const int *lineNumber*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.706.1.7** virtual decaf::lang::exceptions::RuntimeException::~~RuntimeException ( ) throw () [inline, virtual]

**6.706.2 Member Function Documentation**

**6.706.2.1** virtual RuntimeException\* decaf::lang::exceptions::RuntimeException::clone ( ) const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns**

an new **Exception** (p. 1886) that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1889).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**RuntimeException.h**

## 6.707 decaf::security::SecureRandom Class Reference

```
#include <src/main/decaf/security/SecureRandom.h>
```

Inheritance diagram for decaf::security::SecureRandom:

### Public Member Functions

- **SecureRandom** ()  
*Creates a new instance of a secure random number generator that implements the default random number algorithm.*
- **SecureRandom** (const std::vector< unsigned char > &seed)  
*Creates a new instance of a secure random number generator that implements the default random number algorithm.*
- **SecureRandom** (const unsigned char \*seed, int size)  
*Creates a new instance of a secure random number generator that implements the default random number algorithm.*
- virtual ~**SecureRandom** ()
- virtual void **nextBytes** (std::vector< unsigned char > &buf)  
*Modifies the byte array by a random sequence of bytes generated by this random number generator.*

#### Parameters

buf	non-null array to contain the new random bytes
-----	--

#### See also

*next* (p. 3247)

- virtual void **nextBytes** (unsigned char \*buf, int size)  
*Modifies the byte array by a random sequence of bytes generated by this random number generator.*

#### Parameters

buf	non-null array to contain the new random bytes
-----	--

#### See also

*next* (p. 3247)

*Exceptions*

NullPointerException	if <i>buff</i> is <i>NULL</i>
IllegalArgumentException	if <i>size</i> is <i>negative</i>

- virtual void **setSeed** (unsigned long long seed)

*Modifies the seed using linear congruential formula presented in The Art of Computer Programming, Volume 2, Section 3.2.1.*

*Parameters*

seed	<i>the seed that alters the state of the random number generator</i>
------	--

*See also*

*next* (p. 3247)

*Random()* (p. 3247)

*#Random(long)*

- virtual void **setSeed** (const std::vector< unsigned char > &seed)

*Supplements or sets the seed of this secure random number generator; calls to this method never reduces randomness.*

- virtual void **setSeed** (const unsigned char \*seed, int size)

*Supplements or sets the seed of this secure random number generator; calls to this method never reduces randomness.*

**Protected Member Functions**

- virtual int **next** (int bits)

*Answers a pseudo-random uniformly distributed int value of the number of bits specified by the argument bits as described by Donald E.*

*Knuth in The Art of Computer Programming, Volume 2: Seminumerical Algorithms, section 3.2.1.*

*Returns*

*int a pseudo-random generated int number*

*Parameters*

bits	<i>number of bits of the returned value</i>
------	---

*See also*

*nextBytes* (p. 3249)

*nextDouble* (p. 3249)

*nextFloat* (p. 3249)

*nextInt()* (p. 3250)

*nextInt(int)* (p. 3250)

*nextGaussian* (p. 3249)

*nextLong* (p. 3250)

### 6.707.1 Detailed Description

Since

1.0

### 6.707.2 Constructor & Destructor Documentation

#### 6.707.2.1 `decaf::security::SecureRandom::SecureRandom ( )`

Creates a new instance of a secure random number generator that implements the default random number algorithm.

The **SecureRandom** (p. 3424) instance that is created with this constructor is unseeded and can be seeded by calling the `setSeed` method. Calls to `nextBytes` on an unseeded **SecureRandom** (p. 3424) result in the object seeding itself.

#### 6.707.2.2 `decaf::security::SecureRandom::SecureRandom ( const std::vector< unsigned char > & seed )`

Creates a new instance of a secure random number generator that implements the default random number algorithm.

The **SecureRandom** (p. 3424) instance created by this constructor is seeded using the passed byte array.

#### Parameters

<i>seed</i>	The seed bytes to use to seed this secure random number generator.
-------------	--

#### 6.707.2.3 `decaf::security::SecureRandom::SecureRandom ( const unsigned char * seed, int size )`

Creates a new instance of a secure random number generator that implements the default random number algorithm.

The **SecureRandom** (p. 3424) instance created by this constructor is seeded using the passed byte array.

#### Parameters

<i>seed</i>	The seed bytes to use to seed this secure random number generator.
<i>size</i>	The number of bytes in the seed buffer.

#### Exceptions

<i>NullPointerException</i>	if the seed buffer is NULL.
<i>IllegalArgumentException</i>	if the size value is negative.

6.707.2.4 virtual decaf::security::SecureRandom::~SecureRandom ( ) [virtual]

### 6.707.3 Member Function Documentation

6.707.3.1 virtual int decaf::security::SecureRandom::next ( int *bits* ) [protected, virtual]

Answers a pseudo-random uniformly distributed `int` value of the number of bits specified by the argument `bits` as described by Donald E.

Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.2.1.

#### Returns

`int` a pseudo-random generated `int` number

#### Parameters

<i>bits</i>	number of bits of the returned value
-------------	--------------------------------------

#### See also

**nextBytes** (p. 3249)  
**nextDouble** (p. 3249)  
**nextFloat** (p. 3249)  
**nextInt()** (p. 3250)  
**nextInt(int)** (p. 3250)  
**nextGaussian** (p. 3249)  
**nextLong** (p. 3250)

Reimplemented from **decaf::util::Random** (p. 3247).

6.707.3.2 virtual void decaf::security::SecureRandom::nextBytes ( unsigned char \* *buf*, int *size* ) [virtual]

Modifies the byte array by a random sequence of bytes generated by this random number generator.

#### Parameters

<i>buf</i>	non-null array to contain the new random bytes
------------	--

#### See also

**next** (p. 3247)

#### Exceptions

<i>NullPointerException</i>	if <i>buff</i> is NULL
-----------------------------	------------------------

<i>IllegalArgumentException</i>	if size is negative
---------------------------------	---------------------

Reimplemented from **decaf::util::Random** (p. 3248).

**6.707.3.3** `virtual void decaf::security::SecureRandom::nextBytes ( std::vector< unsigned char > & buf ) [virtual]`

Modifies the byte array by a random sequence of bytes generated by this random number generator.

#### Parameters

<i>buf</i>	non-null array to contain the new random bytes
------------	--

#### See also

**next** (p. 3247)

Reimplemented from **decaf::util::Random** (p. 3249).

**6.707.3.4** `virtual void decaf::security::SecureRandom::setSeed ( const std::vector< unsigned char > & seed ) [virtual]`

Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.

#### Parameters

<i>seed</i>	A vector of bytes that is used update the seed of the RNG.
-------------	--

**6.707.3.5** `virtual void decaf::security::SecureRandom::setSeed ( const unsigned char * seed, int size ) [virtual]`

Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.

#### Parameters

<i>seed</i>	The seed bytes to use to seed this secure random number generator.
<i>size</i>	The number of bytes in the seed buffer.

#### Exceptions

<i>NullPointerException</i>	if the seed buffer is NULL.
<i>IllegalArgumentEx-ception</i>	if the size value is negative.



**6.707.3.6** virtual void decaf::security::SecureRandom::setSeed ( unsigned long long *seed* )  
[virtual]

Modifies the seed using linear congruential formula presented in *The Art of Computer Programming, Volume 2*, Section 3.2.1.

#### Parameters

<i>seed</i>	the seed that alters the state of the random number generator
-------------	---

#### See also

**next** (p. 3247)  
**Random()** (p. 3247)  
**#Random(long)**

Reimplemented from **decaf::util::Random** (p. 3251).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**SecureRandom.h**

## 6.708 decaf::internal::security::SecureRandomImpl Class Reference

Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.

```
#include <src/main/decaf/internal/security/unix/SecureRandomImpl.h>
```

Inheritance diagram for decaf::internal::security::SecureRandomImpl:

#### Public Member Functions

- **SecureRandomImpl** ()
- virtual **~SecureRandomImpl** ()
- virtual void **providerSetSeed** (const unsigned char \*seed, int size)  
*Reseed the Random Number Generator; the value given supplements the previous seed value if any instead of replacing it.*
- virtual void **providerNextBytes** (unsigned char \*bytes, int numBytes)  
*Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.*
- virtual unsigned char \* **providerGenerateSeed** (int numBytes)  
*Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.*

- **SecureRandomImpl ()**
- virtual **~SecureRandomImpl ()**
- virtual void **providerSetSeed** (const unsigned char \*seed, int size)  
*Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.*
- virtual void **providerNextBytes** (unsigned char \*bytes, int numBytes)  
*Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.*
- virtual unsigned char \* **providerGenerateSeed** (int numBytes)  
*Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.*

### 6.708.1 Detailed Description

Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources. Secure Random Number Generator for Windows based platforms that attempts to obtain secure bytes with high entropy from known sources.

If the platform does not have a source of secure bytes then the platform random number generator is used if one exists otherwise the Decaf RNG is used as a last resort.

Since

1.0

### 6.708.2 Constructor & Destructor Documentation

6.708.2.1 **decaf::internal::security::SecureRandomImpl::SecureRandomImpl ( )**

6.708.2.2 **virtual decaf::internal::security::SecureRandomImpl::~~SecureRandomImpl ( )**  
 [virtual]

6.708.2.3 **decaf::internal::security::SecureRandomImpl::SecureRandomImpl ( )**

6.708.2.4 **virtual decaf::internal::security::SecureRandomImpl::~~SecureRandomImpl ( )**  
 [virtual]

### 6.708.3 Member Function Documentation

6.708.3.1 **virtual unsigned char\* decaf::internal::security::SecureRandomImpl::providerGenerateSeed ( int numBytes )** [virtual]

Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.

The caller owns the returned array and must delete it.

#### Parameters

<i>numBytes</i>	The number of bytes that should be generated for the new seed array.
-----------------	--

Implements **decaf::security::SecureRandomSpi** (p. 3433).

**6.708.3.2** `virtual unsigned char* decaf::internal::security::SecureRandomImpl::providerGenerateSeed ( int numBytes ) [virtual]`

Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.

The caller owns the returned array and must delete it.

#### Parameters

<i>numBytes</i>	The number of bytes that should be generated for the new seed array.
-----------------	--

Implements **decaf::security::SecureRandomSpi** (p. 3433).

**6.708.3.3** `virtual void decaf::internal::security::SecureRandomImpl::providerNextBytes ( unsigned char * bytes, int numBytes ) [virtual]`

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.

The array must have already been allocated and be of the correct size to prevent segmentation faults.

#### Parameters

<i>bytes</i>	The array that will be filled with random bytes equal to size.
<i>numBytes</i>	The number of bytes to generate and write into the bytes array.

Implements **decaf::security::SecureRandomSpi** (p. 3434).

**6.708.3.4** `virtual void decaf::internal::security::SecureRandomImpl::providerNextBytes ( unsigned char * bytes, int numBytes ) [virtual]`

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.

The array must have already been allocated and be of the correct size to prevent segmentation faults.

#### Parameters

<i>bytes</i>	The array that will be filled with random bytes equal to size.
<i>numBytes</i>	The number of bytes to generate and write into the bytes array.

Implements **decaf::security::SecureRandomSpi** (p. 3434).

**6.708.3.5** `virtual void decaf::internal::security::SecureRandomImpl::providerSetSeed ( const unsigned char * seed, int size ) [virtual]`

Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.

#### Parameters

<i>seed</i>	The array of bytes used to update the generators seed.
<i>size</i>	The size of the passed byte array.

Implements **decaf::security::SecureRandomSpi** (p. 3434).

**6.708.3.6** `virtual void decaf::internal::security::SecureRandomImpl::providerSetSeed ( const unsigned char * seed, int size ) [virtual]`

Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.

#### Parameters

<i>seed</i>	The array of bytes used to update the generators seed.
<i>size</i>	The size of the passed byte array.

Implements **decaf::security::SecureRandomSpi** (p. 3434).

The documentation for this class was generated from the following files:

- src/main/decaf/internal/security/unix/SecureRandomImpl.h
- src/main/decaf/internal/security/windows/SecureRandomImpl.h

## 6.709 decaf::security::SecureRandomSpi Class Reference

Interface class used by Security Service Providers to implement a source of secure random bytes.

```
#include <src/main/decaf/security/SecureRandomSpi.h>
```

Inheritance diagram for decaf::security::SecureRandomSpi:

### Public Member Functions

- **SecureRandomSpi ()**

- virtual `~SecureRandomSpi()`
- virtual void **providerSetSeed** (const unsigned char \*seed, int size)=0  
*Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.*
- virtual void **providerNextBytes** (unsigned char \*bytes, int numBytes)=0  
*Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.*
- virtual unsigned char \* **providerGenerateSeed** (int numBytes)=0  
*Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.*

### 6.709.1 Detailed Description

Interface class used by Security Service Providers to implement a source of secure random bytes.

Since

1.0

### 6.709.2 Constructor & Destructor Documentation

6.709.2.1 `decaf::security::SecureRandomSpi::SecureRandomSpi()`

6.709.2.2 `virtual decaf::security::SecureRandomSpi::~~SecureRandomSpi()` [virtual]

### 6.709.3 Member Function Documentation

6.709.3.1 `virtual unsigned char* decaf::security::SecureRandomSpi::providerGenerateSeed (int numBytes)` [pure virtual]

Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.

The caller owns the returned array and must delete it.

#### Parameters

<i>numBytes</i>	The number of bytes that should be generated for the new seed array.
-----------------	--

Implemented in `decaf::internal::security::SecureRandomImpl` (p. 3431), and `decaf::internal::security::SecureRandomImpl` (p. 3431).

**6.709.3.2** `virtual void decaf::security::SecureRandomSpi::providerNextBytes ( unsigned char * bytes, int numBytes )` [pure virtual]

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.

The array must have already been allocated and be of the correct size to prevent segmentation faults.

#### Parameters

<i>bytes</i>	The array that will be filled with random bytes equal to size.
<i>numBytes</i>	The number of bytes to generate and write into the bytes array.

Implemented in `decaf::internal::security::SecureRandomImpl` (p. 3431), and `decaf::internal::security::SecureRandomImpl` (p. 3431).

**6.709.3.3** `virtual void decaf::security::SecureRandomSpi::providerSetSeed ( const unsigned char * seed, int size )` [pure virtual]

Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.

#### Parameters

<i>seed</i>	The array of bytes used to update the generators seed.
<i>size</i>	The size of the passed byte array.

Implemented in `decaf::internal::security::SecureRandomImpl` (p. 3432), and `decaf::internal::security::SecureRandomImpl` (p. 3432).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/SecureRandomSpi.h`

## 6.710 decaf::util::concurrent::Semaphore Class Reference

A counting semaphore.

```
#include <src/main/decaf/util/concurrent/Semaphore.h>
```

### Public Member Functions

- **Semaphore** (int permits)  
*Creates a **Semaphore** (p. 3434) with the given number of permits and nonfair fairness setting.*
- **Semaphore** (int permits, bool fair)

Creates a **Semaphore** (p. 3434) with the given number of permits and the given fairness setting.

- virtual **~Semaphore** ()
- void **acquire** () throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::RuntimeException )  
*Acquires a permit from this semaphore, blocking until one is available, or the thread is interrupted.*
- void **acquireUninterruptibly** () throw ( decaf::lang::exceptions::RuntimeException )  
*Acquires a permit from this semaphore, blocking until one is available.*
- bool **tryAcquire** () throw ( decaf::lang::exceptions::RuntimeException )  
*Acquires a permit from this semaphore, only if one is available at the time of invocation.*
- bool **tryAcquire** (long long timeout, const **TimeUnit** &unit) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::RuntimeException )  
*Acquires a permit from this semaphore, if one becomes available within the given waiting time and the current thread has not been interrupted.*
- void **release** () throw ( decaf::lang::exceptions::RuntimeException )  
*Releases a permit, returning it to the semaphore.*
- void **acquire** (int permits) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException )  
*Acquires the given number of permits from this semaphore, blocking until all are available, or the thread is interrupted.*
- void **acquireUninterruptibly** (int permits) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException )  
*Acquires the given number of permits from this semaphore, blocking until all are available.*
- bool **tryAcquire** (int permits) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException )  
*Acquires the given number of permits from this semaphore, only if all are available at the time of invocation.*
- bool **tryAcquire** (int permits, long long timeout, const **TimeUnit** &unit) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException )  
*Acquires the given number of permits from this semaphore, if all become available within the given waiting time and the current thread has not been interrupted.*
- void **release** (int permits) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException )

*Releases the given number of permits, returning them to the semaphore.*

- **int availablePermits () const**  
*Returns the current number of permits available in this semaphore.*
- **int drainPermits () throw ( decaf::lang::exceptions::RuntimeException )**  
*Acquires and returns all permits that are immediately available.*
- **bool isFair () const**
- **std::string toString () const**  
*Returns a string identifying this semaphore, as well as its state.*

### 6.710.1 Detailed Description

A counting semaphore. Conceptually, a semaphore maintains a set of permits. Each **acquire()** (p. 3438) blocks if necessary until a permit is available, and then takes it. Each **release()** (p. 3441) adds a permit, potentially releasing a blocking acquirer. However, no actual permit objects are used; the **Semaphore** (p. 3434) just keeps a count of the number available and acts accordingly.

Semaphores are often used to restrict the number of threads than can access some (physical or logical) resource.

```
class Pool { private:
```

```
static const int MAX_AVAILABLE = 100; Semaphore (p. 3434) available;
```

```
std::vector<std::string> items; std::vector<bool> used;
```

```
Mutex (p. 2866) lock;
```

```
public:
```

```
Pool() : available( MAX_AVAILABLE, true ) { used.resize( MAX_AVAILABLE );  
items.resize( MAX_AVAILABLE ); }
```

```
std::string getItem() throws InterruptedException { available.acquire(); return getNextAvailableItem(); }
```

```
void putItem( std::string x ) { if( markAsUnused(x) ) { available.release(); } }
```

```
std::string getNextAvailableItem() {
```

```
synchronized( &lock ) (p. 4713) { for( int i = 0; i < MAX_AVAILABLE; ++i ) { if(  
!used[i] ) { used[i] = true; return items[i]; } }
```

```
return std::string(); // not reached }
```

```
bool markAsUnused( const std::string& item ) { synchronized( &lock ) (p. 4713) {  
for( int i = 0; i < MAX_AVAILABLE; ++i ) { if( item == items[i] ) { if( used[i] ) {  
used[i] = false; return true; } else return false; } } } return false; } };
```

Before obtaining an item each thread must acquire a permit from the semaphore, guaranteeing that an item is available for use. When the thread has finished with the item it is returned back to the pool and a permit is returned to the semaphore, allowing



another thread to acquire that item. Note that no synchronization lock is held when **acquire()** (p. 3438) is called as that would prevent an item from being returned to the pool. The semaphore encapsulates the synchronization needed to restrict access to the pool, separately from any synchronization needed to maintain the consistency of the pool itself.

A semaphore initialized to one, and which is used such that it only has at most one permit available, can serve as a mutual exclusion lock. This is more commonly known as a binary semaphore, because it only has two states: one permit available, or zero permits available. When used in this way, the binary semaphore has the property (unlike many **Lock** (p. 2450) implementations), that the "lock" can be released by a thread other than the owner (as semaphores have no notion of ownership). This can be useful in some specialized contexts, such as deadlock recovery.

The constructor for this class optionally accepts a fairness parameter. When set false, this class makes no guarantees about the order in which threads acquire permits. In particular, barging is permitted, that is, a thread invoking **acquire()** (p. 3438) can be allocated a permit ahead of a thread that has been waiting - logically the new thread places itself at the head of the queue of waiting threads. When fairness is set true, the semaphore guarantees that threads invoking any of the acquire methods are selected to obtain permits in the order in which their invocation of those methods was processed (first-in-first-out; FIFO). Note that FIFO ordering necessarily applies to specific internal points of execution within these methods. So, it is possible for one thread to invoke acquire before another, but reach the ordering point after the other, and similarly upon return from the method. Also note that the untimed tryAcquire methods do not honor the fairness setting, but will take any permits that are available.

Generally, semaphores used to control resource access should be initialized as fair, to ensure that no thread is starved out from accessing a resource. When using semaphores for other kinds of synchronization control, the throughput advantages of non-fair ordering often outweigh fairness considerations.

This class also provides convenience methods to acquire and release multiple permits at a time. Beware of the increased risk of indefinite postponement when these methods are used without fairness set true.

#### Since

1.0

### 6.710.2 Constructor & Destructor Documentation

#### 6.710.2.1 decaf::util::concurrent::Semaphore::Semaphore ( int *permits* )

Creates a **Semaphore** (p. 3434) with the given number of permits and nonfair fairness setting.

#### Parameters

<i>permits</i>	the initial number of permits available. This value may be negative, in which case releases must occur before any acquires will be granted.
----------------	---

### 6.710.2.2 `decaf::util::concurrent::Semaphore::Semaphore ( int permits, bool fair )`

Creates a **Semaphore** (p. 3434) with the given number of permits and the given fairness setting.

#### Parameters

<i>permits</i>	the initial number of permits available. This value may be negative, in which case releases must occur before any acquires will be granted.
<i>fair</i>	true if this semaphore will guarantee first-in first-out granting of permits under contention, else false

### 6.710.2.3 `virtual decaf::util::concurrent::Semaphore::~Semaphore ( ) [virtual]`

## 6.710.3 Member Function Documentation

### 6.710.3.1 `void decaf::util::concurrent::Semaphore::acquire ( ) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::RuntimeException )`

Acquires a permit from this semaphore, blocking until one is available, or the thread is interrupted.

Acquires a permit, if one is available and returns immediately, reducing the number of available permits by one.

If no permit is available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

\* Some other thread invokes the **release()** (p. 3441) method for this semaphore and the current thread is next to be assigned a permit; or \* Some other thread interrupts the current thread.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while waiting for a permit,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared.

#### Exceptions

<i>InterruptedException</i>	- if the current thread is interrupted.
<i>RuntimeException</i>	if an unexpected error occurs while acquiring the <b>Semaphore</b> (p. 3434).

**6.710.3.2** `void decaf::util::concurrent::Semaphore::acquire ( int permits )  
 throw ( decaf::lang::exceptions::InterruptedException,  
 decaf::lang::exceptions::IllegalArgumentException,  
 decaf::lang::exceptions::RuntimeException )`

Acquires the given number of permits from this semaphore, blocking until all are available, or the thread is interrupted.

Acquires the given number of permits, if they are available, and returns immediately, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

\* Some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this request; or \* Some other thread interrupts the current thread.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while waiting for a permit,

then InterruptedException is thrown and the current thread's interrupted status is cleared. Any permits that were to be assigned to this thread are instead assigned to other threads trying to acquire permits, as if permits had been made available by a call to **release()** (p. 3441).

#### Parameters

<i>permits</i>	the number of permits to acquire.
----------------	-----------------------------------

#### Exceptions

<i>InterruptedException</i>	if the current thread is interrupted.
<i>IllegalArgumentException</i>	if the permits argument is negative.
<i>RuntimeException</i>	if an unexpected error occurs while acquiring the <b>Semaphore</b> (p. 3434).

**6.710.3.3** `void decaf::util::concurrent::Semaphore::acquireUninterruptibly ( ) throw ( decaf::lang::exceptions::RuntimeException )`

Acquires a permit from this semaphore, blocking until one is available.

Acquires a permit, if one is available and returns immediately, reducing the number of available permits by one.

If no permit is available then the current thread becomes disabled for thread scheduling purposes and lies dormant until some other thread invokes the **release()** (p. 3441) method for this semaphore and the current thread is next to be assigned a permit.

If the current thread is interrupted while waiting for a permit then it will continue to wait, but the time at which the thread is assigned a permit may change compared to the

time it would have received the permit had no interruption occurred. When the thread does return from this method its interrupt status will be set.

### Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while acquiring the <b>Semaphore</b> (p. 3434).
-------------------------	---

**6.710.3.4** `void decaf::util::concurrent::Semaphore::acquireUninterruptibly ( int permits ) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException )`

Acquires the given number of permits from this semaphore, blocking until all are available.

Acquires the given number of permits, if they are available, and returns immediately, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this request.

If the current thread is interrupted while waiting for permits then it will continue to wait and its position in the queue is not affected. When the thread does return from this method its interrupt status will be set.

### Parameters

<i>permits</i>	the number of permits to acquire.
----------------	-----------------------------------

### Exceptions

<i>IllegalArgumentException</i>	if the permits argument is negative.
<i>RuntimeException</i>	if an unexpected error occurs while acquiring the <b>Semaphore</b> (p. 3434).

**6.710.3.5** `int decaf::util::concurrent::Semaphore::availablePermits ( ) const`

Returns the current number of permits available in this semaphore.

This method is typically used for debugging and testing purposes.

### Returns

the number of permits available in this semaphore

**6.710.3.6** `int decaf::util::concurrent::Semaphore::drainPermits ( ) throw ( decaf::lang::exceptions::RuntimeException )`

Acquires and returns all permits that are immediately available.

#### Returns

the number of permits acquired

#### Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while draining the <b>Semaphore</b> (p. 3434).
-------------------------	--

**6.710.3.7** `bool decaf::util::concurrent::Semaphore::isFair ( ) const`

#### Returns

true if this semaphore has fairness set true

**6.710.3.8** `void decaf::util::concurrent::Semaphore::release ( ) throw ( decaf::lang::exceptions::RuntimeException )`

Releases a permit, returning it to the semaphore.

Releases a permit, increasing the number of available permits by one. If any threads are trying to acquire a permit, then one is selected and given the permit that was just released. That thread is (re)enabled for thread scheduling purposes.

There is no requirement that a thread that releases a permit must have acquired that permit by calling **acquire()** (p. 3438). Correct usage of a semaphore is established by programming convention in the application.

#### Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while releasing the <b>Semaphore</b> (p. 3434).
-------------------------	---

**6.710.3.9** `void decaf::util::concurrent::Semaphore::release ( int permits ) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException )`

Releases the given number of permits, returning them to the semaphore.

Releases the given number of permits, increasing the number of available permits by that amount. If any threads are trying to acquire permits, then one is selected and given the permits that were just released. If the number of available permits satisfies that thread's request then that thread is (re)enabled for thread scheduling purposes;

otherwise the thread will wait until sufficient permits are available. If there are still permits available after this thread's request has been satisfied, then those permits are assigned in turn to other threads trying to acquire permits.

#### Parameters

<i>permits</i>	the number of permits to release
----------------	----------------------------------

#### Exceptions

<i>IllegalArgumentException</i>	if the permits argument is negative.
<i>RuntimeException</i>	if an unexpected error occurs while releasing the <b>Semaphore</b> (p. 3434).

#### 6.710.3.10 `std::string decaf::util::concurrent::Semaphore::toString ( ) const`

Returns a string identifying this semaphore, as well as its state.

The state, in brackets, includes the String "Permits =" followed by the number of permits.

#### Returns

a string identifying this semaphore, as well as its state

#### 6.710.3.11 `bool decaf::util::concurrent::Semaphore::tryAcquire ( int permits, long long timeout, const TimeUnit & unit ) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException )`

Acquires the given number of permits from this semaphore, if all become available within the given waiting time and the current thread has not been interrupted.

Acquires the given number of permits, if they are available and returns immediately, with the value true, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

\* Some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this request; or \* Some other thread interrupts the current thread; or \* The specified waiting time elapses.

If the permits are acquired then the value true is returned.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while waiting to acquire the permits,

then InterruptedException is thrown and the current thread's interrupted status is cleared. Any permits that were to be assigned to this thread, are instead assigned to other threads trying to acquire permits, as if the permits had been made available by a call to **release()** (p. 3441).

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all. Any permits that were to be assigned to this thread, are instead assigned to other threads trying to acquire permits, as if the permits had been made available by a call to **release()** (p. 3441).

#### Parameters

<i>permits</i>	the number of permits to acquire
<i>timeout</i>	the maximum amount of time to wait to acquire the permits.
<i>unit</i>	the units that the timeout param represents.

#### Returns

true if all permits were acquired and false if the waiting time elapsed before all permits were acquired

#### Exceptions

<i>IllegalArgumentException</i>	if the permits argument is negative.
<i>RuntimeException</i>	if an unexpected error occurs while acquiring the <b>Semaphore</b> (p. 3434).

**6.710.3.12** `bool decaf::util::concurrent::Semaphore::tryAcquire ( long long timeout, const TimeUnit & unit ) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::RuntimeException )`

Acquires a permit from this semaphore, if one becomes available within the given waiting time and the current thread has not been interrupted.

Acquires a permit, if one is available and returns immediately, with the value true, reducing the number of available permits by one.

If no permit is available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

\* Some other thread invokes the **release()** (p. 3441) method for this semaphore and the current thread is next to be assigned a permit; or \* Some other thread interrupts the current thread; or \* The specified waiting time elapses.

If a permit is acquired then the value true is returned.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while waiting to acquire a permit,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

#### Parameters

<i>timeout</i>	the maximum time to wait for a permit
<i>unit</i>	the time unit of the timeout argument

#### Returns

true if a permit was acquired and false if the waiting time elapsed before a permit was acquired

#### Exceptions

<i>InterruptedException</i>	if the current thread is interrupted.
<i>RuntimeException</i>	if an unexpected error occurs while acquiring the <b>Semaphore</b> (p. 3434).

#### 6.710.3.13 `bool decaf::util::concurrent::Semaphore::tryAcquire ( ) throw ( decaf::lang::exceptions::RuntimeException )`

Acquires a permit from this semaphore, only if one is available at the time of invocation.

Acquires a permit, if one is available and returns immediately, with the value true, reducing the number of available permits by one.

If no permit is available then this method will return immediately with the value false.

Even when this semaphore has been set to use a fair ordering policy, a call to **tryAcquire()** (p. 3444) will immediately acquire a permit if one is available, whether or not other threads are currently waiting. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting, then use **tryAcquire(0, TimeUnit.SECONDS** (p. 3930)) which is almost equivalent (it also detects interruption).

#### Returns

true if a permit was acquired and false otherwise

#### Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while acquiring the <b>Semaphore</b> (p. 3434).
-------------------------	---



**6.710.3.14** `bool decaf::util::concurrent::Semaphore::tryAcquire ( int permits )  
 throw ( decaf::lang::exceptions::IllegalArgumentException,  
 decaf::lang::exceptions::RuntimeException )`

Acquires the given number of permits from this semaphore, only if all are available at the time of invocation.

Acquires the given number of permits, if they are available, and returns immediately, with the value true, reducing the number of available permits by the given amount.

If insufficient permits are available then this method will return immediately with the value false and the number of available permits is unchanged.

Even when this semaphore has been set to use a fair ordering policy, a call to tryAcquire will immediately acquire a permit if one is available, whether or not other threads are currently waiting. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting, then use tryAcquire(permits, 0, **TimeUnit.SECONDS** (p. 3930)) which is almost equivalent (it also detects interruption).

#### Parameters

<i>permits</i>	the number of permits to acquire
----------------	----------------------------------

#### Returns

true if the permits were acquired and false otherwise.

#### Exceptions

<i>IllegalArgumentException</i>	if the permits argument is negative.
<i>RuntimeException</i>	if an unexpected error occurs while acquiring the <b>Semaphore</b> (p. 3434).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Semaphore.h**

## 6.711 activemq::cmsutil::CmsTemplate::SendExecutor Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for activemq::cmsutil::CmsTemplate::SendExecutor:

## Public Member Functions

- **SendExecutor** (**MessageCreator** \*messageCreator, **CmsTemplate** \*parent)
- virtual **~SendExecutor** ()
- virtual void **doInCms** (**cms::Session** \*session, **cms::MessageProducer** \*producer) throw ( **cms::CMSEException** )

*Execute an action given a session and producer.*

## Protected Member Functions

- **SendExecutor** (const **SendExecutor** &)
- **SendExecutor** & **operator=** (const **SendExecutor** &)

## 6.711.1 Constructor & Destructor Documentation

6.711.1.1 **activemq::cmsutil::CmsTemplate::SendExecutor::SendExecutor** ( const **SendExecutor** & ) [inline, protected]

6.711.1.2 **activemq::cmsutil::CmsTemplate::SendExecutor::SendExecutor** ( **MessageCreator** \* *messageCreator*, **CmsTemplate** \* *parent* ) [inline]

6.711.1.3 virtual **activemq::cmsutil::CmsTemplate::SendExecutor::~~SendExecutor** ( ) [inline, virtual]

## 6.711.2 Member Function Documentation

6.711.2.1 virtual void **activemq::cmsutil::CmsTemplate::SendExecutor::doInCms** ( **cms::Session** \* *session*, **cms::MessageProducer** \* *producer* ) throw ( **cms::CMSEException** ) [inline, virtual]

Execute an action given a session and producer.

## Parameters

<i>session</i>	the CMS Session
<i>producer</i>	the CMS Producer

## Exceptions

<b>cms::CMSEException</b> (p. 1190)	if thrown by CMS API methods
--	------------------------------

Implements **activemq::cmsutil::ProducerCallback** (p. 3154).

### 6.711.2.2 SendExecutor& activemq::cmsutil::CmsTemplate::SendExecutor::operator= ( const SendExecutor & ) [inline, protected]

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/CmsTemplate.h

## 6.712 decaf::net::ServerSocket Class Reference

This class implements server sockets.

```
#include <src/main/decaf/net/ServerSocket.h>
```

Inheritance diagram for decaf::net::ServerSocket:

### Public Member Functions

- **ServerSocket** ()  
*Creates a non-bound server socket.*
- **ServerSocket** (int port) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Creates a new **ServerSocket** (p. 3447) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- **ServerSocket** (int port, int backlog) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Creates a new **ServerSocket** (p. 3447) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- **ServerSocket** (int port, int backlog, const **InetAddress** \*address) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Creates a new **ServerSocket** (p. 3447) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- virtual ~**ServerSocket** ()  
*Releases socket handle if **close()** (p. 3453) hasn't been called.*
- virtual void **bind** (const std::string &host, int port) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Bind and listen to given local IP address and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.*
- virtual void **bind** (const std::string &host, int port, int backlog) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )

*Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.*

- virtual **Socket \* accept** () throw ( decaf::io::IOException )  
*Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 3447), the caller blocks until a connection is made.*
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes the server socket, causing any Threads blocked on an accept call to throw an Exception.*
- virtual bool **isClosed** () const
- virtual bool **isBound** () const
- virtual int **getReceiveBufferSize** () const throw ( SocketException )  
*Gets the receive buffer size for this socket, SO\_RCVBUF.*
- virtual void **setReceiveBufferSize** (int size) throw ( SocketException, decaf::lang::exceptions::IllegalArgument )  
*Sets the receive buffer size for this socket, SO\_RCVBUF.*
- virtual bool **getReuseAddress** () const throw ( SocketException )  
*Gets the reuse address flag, SO\_REUSEADDR.*
- virtual void **setReuseAddress** (bool reuse) throw ( SocketException )  
*Sets the reuse address flag, SO\_REUSEADDR.*
- virtual int **getSoTimeout** () const throw ( SocketException )  
*Gets the timeout for socket operations, SO\_TIMEOUT.*
- virtual void **setSoTimeout** (int timeout) throw ( SocketException, decaf::lang::exceptions::IllegalArgument )  
*Sets the timeout for socket operations, SO\_TIMEOUT.*
- virtual int **getLocalPort** () const  
*Gets the port number on the Local machine that this **ServerSocket** (p. 3447) is bound to.*
- virtual std::string **toString** () const

### Static Public Member Functions

- static void **setSocketImplFactory** (SocketImplFactory \*factory) throw ( decaf::io::IOException, decaf::net::SocketException )  
*Sets the instance of a **SocketImplFactory** (p. 3644) that the **ServerSocket** (p. 3447) class should use when new instances of this class are created.*

## Protected Member Functions

- **ServerSocket** (**SocketImpl** \*impl)

*Creates a **ServerSocket** (p. 3447) wrapping the provided **SocketImpl** (p. 3635) instance, this **Socket** (p. 3607) is considered unconnected.*

- virtual void **implAccept** (**Socket** \*socket) throw ( decaf::io::IOException )

*Virtual method that allows a **ServerSocket** (p. 3447) subclass to override the accept call and provide its own **SocketImpl** (p. 3635) for the socket.*

- virtual int **getDefaultBacklog** ()

*Allows a subclass to override what is considered the default backlog.*

- void **checkClosed** () const throw ( decaf::io::IOException )

- void **ensureCreated** () const throw ( decaf::io::IOException )

- void **setupSocketImpl** (int port, int backlog, const **InetAddress** \*ifAddress)

### 6.712.1 Detailed Description

This class implements server sockets. A server socket waits for requests to come in over the network.

The actual work of the server socket is performed by an instance of the **SocketImpl** (p. 3635) class. An application can change the socket factory that creates the socket implementation to configure itself to create sockets of a particular type.

#### Since

1.0

### 6.712.2 Constructor & Destructor Documentation

#### 6.712.2.1 decaf::net::ServerSocket::ServerSocket ( )

Creates a non-bound server socket.

#### 6.712.2.2 decaf::net::ServerSocket::ServerSocket ( int port ) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )

Creates a new **ServerSocket** (p. 3447) bound to the specified port, if the value of port is 0, then any free port is chosen.

When this constructor is called the size of the backlog queue is set at 50, connections that arrive after the backlog has been reached are refused.

If a **SocketImplFactory** (p. 3644) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 3635) is created.

**Parameters**

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3447) to.
-------------	--

**Exceptions**

<i>IOException</i>	if there is an I/O error while performing this operation.
<i>IllegalArgumentException</i>	if the port value is negative or greater than 65535.

**6.712.2.3** `decaf::net::ServerSocket::ServerSocket ( int port,  
int backlog ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::IllegalArgumentException )`

Creates a new **ServerSocket** (p. 3447) bound to the specified port, if the value of port is 0, then any free port is chosen.

When this constructor is called the size of the backlog queue is set at backlog, connections that arrive after the backlog has been reached are refused. If backlog is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 3644) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 3635) is created.

**Parameters**

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3447) to.
<i>backlog</i>	The the number of incoming connection attempts to queue before connections are refused.

**Exceptions**

<i>IOException</i>	if there is an I/O error while performing this operation.
<i>IllegalArgumentException</i>	if the port value is negative or greater than 65535.

**6.712.2.4** `decaf::net::ServerSocket::ServerSocket ( int port, int backlog, const  
InetAddress * address ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::IllegalArgumentException )`

Creates a new **ServerSocket** (p. 3447) bound to the specified port, if the value of port is 0, then any free port is chosen.

If the value of the ifAddress is empty or NULL then the ANY address is used.

When this constructor is called the size of the backlog queue is set at backlog, connections that arrive after the backlog has been reached are refused. If backlog is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 3644) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 3635) is created.

**Parameters**

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3447) to.
<i>backlog</i>	The the number of incoming connection attempts to queue before connections are refused.
<i>ifAddress</i>	The IP Address to bind to on the local machine.

**Exceptions**

<i>IOException</i>	if there is an I/O error while performing this operation.
<i>IllegalArgumentEx-ception</i>	if the port value is negative or greater than 65535.

**6.712.2.5 virtual decaf::net::ServerSocket::~ServerSocket ( ) [virtual]**

Releases socket handle if **close()** (p. 3453) hasn't been called.

**6.712.2.6 decaf::net::ServerSocket::ServerSocket ( SocketImpl \* impl ) [protected]**

Creates a **ServerSocket** (p. 3447) wrapping the provided **SocketImpl** (p. 3635) instance, this **Socket** (p. 3607) is considered unconnected.

The **ServerSocket** (p. 3447) class takes ownership of this **SocketImpl** (p. 3635) pointer and will delete it when the **Socket** (p. 3607) class is destroyed.

**Parameters**

<i>impl</i>	The <b>SocketImpl</b> (p. 3635) instance to wrap.
-------------	---

**Exceptions**

<i>NullPointerException</i>	if the passed <b>SocketImpl</b> (p. 3635) is Null.
-----------------------------	--

**6.712.3 Member Function Documentation****6.712.3.1 virtual Socket\* decaf::net::ServerSocket::accept ( ) throw ( decaf::io::IOException ) [virtual]**

Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 3447), the caller blocks until a connection is made.

If the SO\_TIMEOUT option is set this method could throw a **SocketTimeoutException** (p. 3650) if the operation times out.

**Returns**

a new **Socket** (p. 3607) object pointer. Never returns NULL, the returned pointer is owned by the caller and must be explicitly freed by them.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while binding the socket.
<i>SocketException</i> (p. 3627)	if an error occurs while blocking on the accept call.
<i>SocketTimeoutException</i> (p. 3650)	if the SO_TIMEOUT option was used and the accept timed out.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2932).

**6.712.3.2** `virtual void decaf::net::ServerSocket::bind ( const std::string & host, int port, int backlog ) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException ) [virtual]`

Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.

If the backlog is greater than zero it will be used instead of the default value, otherwise the default value is used and no error is generated.

**Parameters**

<i>host</i>	The IP address or host name.
<i>port</i>	The TCP port between 1..65535.
<i>backlog</i>	The size of listen backlog.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while binding the socket.
<i>IllegalArgumentException</i>	if the parameters are not valid.

**6.712.3.3** `virtual void decaf::net::ServerSocket::bind ( const std::string & host, int port ) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException ) [virtual]`

Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.

**Parameters**

<i>host</i>	The IP address or host name.
<i>port</i>	The TCP port between 1..65535.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while binding the socket.
--------------------	--



<i>IllegalArgumentException</i>	if the parameters are not valid.
---------------------------------	----------------------------------

**6.712.3.4** void decaf::net::ServerSocket::checkClosed ( ) const throw ( decaf::io::IOException ) [protected]

**6.712.3.5** virtual void decaf::net::ServerSocket::close ( ) throw ( decaf::io::IOException ) [virtual]

Closes the server socket, causing any Threads blocked on an accept call to throw an Exception.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

**6.712.3.6** void decaf::net::ServerSocket::ensureCreated ( ) const throw ( decaf::io::IOException ) [protected]

**6.712.3.7** virtual int decaf::net::ServerSocket::getDefaultBacklog ( ) [protected, virtual]

Allows a subclass to override what is considered the default backlog.

#### Returns

the default backlog for connections.

**6.712.3.8** virtual int decaf::net::ServerSocket::getLocalPort ( ) const [virtual]

Gets the port number on the Local machine that this **ServerSocket** (p.3447) is bound to.

#### Returns

the port number of this machine that is bound, if not bound returns -1.

**6.712.3.9** virtual int decaf::net::ServerSocket::getReceiveBufferSize ( ) const throw ( SocketException ) [virtual]

Gets the receive buffer size for this socket, SO\_RCVBUF.

This is the buffer used by the underlying platform socket to buffer received data.

**Returns**

the receive buffer size in bytes.

**Exceptions**

<i><b>SocketException</b></i> (p. 3627)	if the operation fails.
--	-------------------------

**6.712.3.10** `virtual bool decaf::net::ServerSocket::getReuseAddress ( ) const throw ( SocketException )` [virtual]

Gets the reuse address flag, SO\_REUSEADDR.

**Returns**

True if the address can be reused.

**Exceptions**

<i><b>SocketException</b></i> (p. 3627)	if the operation fails.
--	-------------------------

**6.712.3.11** `virtual int decaf::net::ServerSocket::getSoTimeout ( ) const throw ( SocketException )` [virtual]

Gets the timeout for socket operations, SO\_TIMEOUT.

**Returns**

The timeout in milliseconds for socket operations.

**Exceptions**

<i><b>SocketException</b></i> (p. 3627)	Thrown if unable to retrieve the information.
--	---

**6.712.3.12** `virtual void decaf::net::ServerSocket::implAccept ( Socket * socket ) throw ( decaf::io::IOException )` [protected, virtual]

Virtual method that allows a **ServerSocket** (p. 3447) subclass to override the accept call and provide its own **SocketImpl** (p. 3635) for the socket.

**Parameters**

<i>socket</i>	The socket object whose <b>SocketImpl</b> (p. 3635) should be used for the accept call.
---------------	---

**Exceptions**

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

**6.712.3.13** virtual bool decaf::net::ServerSocket::isBound ( ) const [virtual]

**Returns**

true if the server socket is bound.

**6.712.3.14** virtual bool decaf::net::ServerSocket::isClosed ( ) const [virtual]

**Returns**

true if the close method has been called on the **ServerSocket** (p. 3447).

**6.712.3.15** virtual void decaf::net::ServerSocket::setReceiveBufferSize ( int *size* ) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException ) [virtual]

Sets the receive buffer size for this socket, SO\_RCVBUF.

**Parameters**

<i>size</i>	Number of bytes to set the receive buffer to.
-------------	---

**Exceptions**

<i>SocketException</i> (p. 3627)	if the operation fails.
<i>IllegalArgumentException</i>	if the value is zero or negative.

**6.712.3.16** virtual void decaf::net::ServerSocket::setReuseAddress ( bool *reuse* ) throw ( SocketException ) [virtual]

Sets the reuse address flag, SO\_REUSEADDR.

**Parameters**

<i>reuse</i>	If true, sets the flag.
--------------	-------------------------

**Exceptions**

<i>SocketException</i> (p. 3627)	if the operation fails.
-------------------------------------	-------------------------

**6.712.3.17** `static void decaf::net::ServerSocket::setSocketImplFactory ( SocketImplFactory * factory ) throw ( decaf::io::IOException, decaf::net::SocketException )`  
`[static]`

Sets the instance of a **SocketImplFactory** (p. 3644) that the **ServerSocket** (p. 3447) class should use when new instances of this class are created.

This method is only allowed to be used once during the lifetime of the application.

#### Parameters

<i>factory</i>	The instance of a <b>SocketImplFactory</b> (p. 3644) to use when new <b>SocketImpl</b> (p. 3635) objects are created.
----------------	---

#### Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
<i>SocketException</i> (p. 3627)	if this method has already been called with a valid factory.

**6.712.3.18** `virtual void decaf::net::ServerSocket::setSoTimeout ( int timeout ) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )`  
`[virtual]`

Sets the timeout for socket operations, SO\_TIMEOUT.

A value of zero indicates that timeout is infinite for operations on this socket.

#### Parameters

<i>timeout</i>	The timeout in milliseconds for socket operations.
----------------	--

#### Exceptions

<i>SocketException</i> (p. 3627)	Thrown if unable to set the information.
<i>IllegalArgumentException</i>	if the timeout value is negative.

**6.712.3.19** `void decaf::net::ServerSocket::setupSocketImpl ( int port, int backlog, const InetAddress * ifAddress )` `[protected]`

**6.712.3.20** `virtual std::string decaf::net::ServerSocket::toString ( ) const` `[virtual]`

#### Returns

a string representing this **ServerSocket** (p. 3447).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ServerSocket.h`

## 6.713 decaf::net::ServerSocketFactory Class Reference

Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.

```
#include <src/main/decaf/net/ServerSocketFactory.h>
```

Inheritance diagram for decaf::net::ServerSocketFactory:

### Public Member Functions

- virtual `~ServerSocketFactory ()`
- virtual `ServerSocket * createServerSocket ()`  
*Create a new **ServerSocket** (p. 3447) that is unbound.*
- virtual `ServerSocket * createServerSocket (int port)=0`  
*Create a new **ServerSocket** (p. 3447) that is bound to the given port.*
- virtual `ServerSocket * createServerSocket (int port, int backlog)=0`  
*Create a new **ServerSocket** (p. 3447) that is bound to the given port.*
- virtual `ServerSocket * createServerSocket (int port, int backlog, const InetAddress *address)=0`  
*Create a new **ServerSocket** (p. 3447) that is bound to the given port.*

### Static Public Member Functions

- static `ServerSocketFactory * getDefault ()`  
*Returns the Default **ServerSocket** (p. 3447) factory, the pointer is owned by the Decaf runtime and should not be deleted by the caller.*

### Protected Member Functions

- `ServerSocketFactory ()`

#### 6.713.1 Detailed Description

Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.

**Since**

1.0

**6.713.2 Constructor & Destructor Documentation****6.713.2.1** `decaf::net::ServerSocketFactory::ServerSocketFactory ( )` [protected]**6.713.2.2** `virtual decaf::net::ServerSocketFactory::~ServerSocketFactory ( )` [virtual]**6.713.3 Member Function Documentation****6.713.3.1** `virtual ServerSocket* decaf::net::ServerSocketFactory::createServerSocket ( )`  
[virtual]Create a new **ServerSocket** (p. 3447) that is unbound.The **ServerSocket** (p. 3447) will have been configured with the defaults from the factory.**Returns**new **ServerSocket** (p. 3447) pointer that is owned by the caller.**Exceptions***IOException* if the **ServerSocket** (p. 3447) cannot be created for some reason.Reimplemented in **decaf::internal::net::DefaultServerSocketFactory** (p. 1736), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1747), and **decaf::internal::net::ssl::openssl::DefaultSSLServerSocketFactory** (p. 2938).**6.713.3.2** `virtual ServerSocket* decaf::net::ServerSocketFactory::createServerSocket ( int port )` [pure virtual]Create a new **ServerSocket** (p. 3447) that is bound to the given port.The **ServerSocket** (p. 3447) will have been configured with the defaults from the factory.**Parameters***port* The port to bind the **ServerSocket** (p. 3447) to.**Returns**new **ServerSocket** (p. 3447) pointer that is owned by the caller.**Exceptions***IOException* if the **ServerSocket** (p. 3447) cannot be created for some reason.

Implemented in **decaf::internal::net::DefaultServerSocketFactory** (p. 1738), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1747), and **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2938).

**6.713.3.3** `virtual ServerSocket* decaf::net::ServerSocketFactory::createServerSocket ( int port, int backlog, const InetAddress * address ) [pure virtual]`

Create a new **ServerSocket** (p. 3447) that is bound to the given port.

The **ServerSocket** (p. 3447) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3447) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 3447) will listen on all interfaces.

#### Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3447) to.
<i>backlog</i>	The number of pending connect request the <b>ServerSocket</b> (p. 3447) can queue.
<i>address</i>	The address of the interface on the local machine to bind to.

#### Returns

new **ServerSocket** (p. 3447) pointer that is owned by the caller.

#### Exceptions

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3447) cannot be created for some reason.
--------------------	---

Implemented in **decaf::internal::net::DefaultServerSocketFactory** (p. 1737), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1748), and **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2939).

**6.713.3.4** `virtual ServerSocket* decaf::net::ServerSocketFactory::createServerSocket ( int port, int backlog ) [pure virtual]`

Create a new **ServerSocket** (p. 3447) that is bound to the given port.

The **ServerSocket** (p. 3447) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3447) will use the specified connection backlog setting.

#### Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3447) to.
<i>backlog</i>	The number of pending connect request the <b>ServerSocket</b> (p. 3447) can queue.

#### Returns

new **ServerSocket** (p. 3447) pointer that is owned by the caller.

**Exceptions**

*IOException* if the **ServerSocket** (p. 3447) cannot be created for some reason.

Implemented in **decaf::internal::net::DefaultServerSocketFactory** (p. 1737), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1748), and **decaf::internal::net::ssl::openssl::DefaultSSLServerSocketFactory** (p. 2939).

**6.713.3.5 static ServerSocketFactory\* decaf::net::ServerSocketFactory::getDefault ( )**  
[static]

Returns the Default **ServerSocket** (p. 3447) factory, the pointer is owned by the Decaf runtime and should not be deleted by the caller.

Only one default **ServerSocketFactory** (p. 3457) exists for the lifetime of the Application.

**Returns**

the default **ServerSocketFactory** (p. 3457) for this application.

Reimplemented in **decaf::net::ssl::SSLServerSocketFactory** (p. 3669).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**ServerSocketFactory.h**

**6.714 cms::Session Class Reference**

A **Session** (p. 3460) object is a single-threaded context for producing and consuming messages.

```
#include <src/main/cms/Session.h>
```

Inheritance diagram for cms::Session:

**Public Types**

- enum **AcknowledgeMode** {  
    **AUTO\_ACKNOWLEDGE**, **DUPS\_OK\_ACKNOWLEDGE**, **CLIENT\_ACKNOWLEDGE**,  
    **SESSION\_TRANSACTED**,  
    **INDIVIDUAL\_ACKNOWLEDGE** }

**Public Member Functions**

- virtual **~Session** ()



- virtual void **close** ()=0 throw ( CMSEException )  
*Closes this session as well as any active child consumers or producers.*
- virtual void **commit** ()=0 throw ( CMSEException )  
*Commits all messages done in this transaction and releases any locks currently held.*
- virtual void **rollback** ()=0 throw ( CMSEException )  
*Rolls back all messages done in this transaction and releases any locks currently held.*
- virtual void **recover** ()=0 throw ( CMSEException )  
*Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.*
- virtual **MessageConsumer** \* **createConsumer** (const **Destination** \*destination)=0 throw ( CMSEException )  
*Creates a **MessageConsumer** (p. 2674) for the specified destination.*
- virtual **MessageConsumer** \* **createConsumer** (const **Destination** \*destination, const std::string &selector)=0 throw ( CMSEException )  
*Creates a **MessageConsumer** (p. 2674) for the specified destination, using a message selector.*
- virtual **MessageConsumer** \* **createConsumer** (const **Destination** \*destination, const std::string &selector, bool noLocal)=0 throw ( CMSEException )  
*Creates a **MessageConsumer** (p. 2674) for the specified destination, using a message selector.*
- virtual **MessageConsumer** \* **createDurableConsumer** (const **Topic** \*destination, const std::string &name, const std::string &selector, bool noLocal=false)=0 throw ( CMSEException )  
*Creates a durable subscriber to the specified topic, using a **Message** (p. 2614) selector.*
- virtual **MessageProducer** \* **createProducer** (const **Destination** \*destination)=0 throw ( CMSEException )  
*Creates a **MessageProducer** (p. 2809) to send messages to the specified destination.*
- virtual **QueueBrowser** \* **createBrowser** (const **cms::Queue** \*queue)=0 throw ( CMSEException )  
*Creates a new **QueueBrowser** (p. 3243) to peek at Messages on the given **Queue** (p. 3238).*
- virtual **QueueBrowser** \* **createBrowser** (const **cms::Queue** \*queue, const std::string &selector)=0 throw ( CMSEException )  
*Creates a new **QueueBrowser** (p. 3243) to peek at Messages on the given **Queue** (p. 3238).*

- virtual **Queue** \* **createQueue** (const std::string &queueName)=0 throw ( CMSEException )  
*Creates a queue identity given a **Queue** (p. 3238) name.*
- virtual **Topic** \* **createTopic** (const std::string &topicName)=0 throw ( CMSEException )  
*Creates a topic identity given a **Queue** (p. 3238) name.*
- virtual **TemporaryQueue** \* **createTemporaryQueue** ()=0 throw ( CMSEException )  
*Creates a **TemporaryQueue** (p. 3871) object.*
- virtual **TemporaryTopic** \* **createTemporaryTopic** ()=0 throw ( CMSEException )  
*Creates a **TemporaryTopic** (p. 3873) object.*
- virtual **Message** \* **createMessage** ()=0 throw ( CMSEException )  
*Creates a new **Message** (p. 2614).*
- virtual **BytesMessage** \* **createBytesMessage** ()=0 throw ( CMSEException )  
*Creates a **BytesMessage** (p. 1079).*
- virtual **BytesMessage** \* **createBytesMessage** (const unsigned char \*bytes, int bytesSize)=0 throw ( CMSEException )  
*Creates a **BytesMessage** (p. 1079) and sets the payload to the passed value.*
- virtual **StreamMessage** \* **createStreamMessage** ()=0 throw ( CMSEException )  
*Creates a new **StreamMessage** (p. 3760).*
- virtual **TextMessage** \* **createTextMessage** ()=0 throw ( CMSEException )  
*Creates a new **TextMessage** (p. 3874).*
- virtual **TextMessage** \* **createTextMessage** (const std::string &text)=0 throw ( CMSEException )  
*Creates a new **TextMessage** (p. 3874) and set the text to the value given.*
- virtual **MapMessage** \* **createMapMessage** ()=0 throw ( CMSEException )  
*Creates a new **MapMessage** (p. 2551).*
- virtual **AcknowledgeMode** **getAcknowledgeMode** () const =0 throw ( CMSEException )  
*Returns the acknowledgment mode of the session.*
- virtual bool **isTransacted** () const =0 throw ( CMSEException )  
*Gets if the Sessions is a Transacted **Session** (p. 3460).*

- virtual void **unsubscribe** (const std::string &name)=0 throw ( CMSEException )

*Unsubscribes a durable subscription that has been created by a client.*

### 6.714.1 Detailed Description

A **Session** (p. 3460) object is a single-threaded context for producing and consuming messages. A session serves several purposes:

- It is a factory for its message producers and consumers.
- It supplies provider-optimized message factories.
- It is a factory for TemporaryTopics and TemporaryQueues.
- It provides a way to create **Queue** (p. 3238) or **Topic** (p. 3930) objects for those clients that need to dynamically manipulate provider-specific destination names.
- It supports a single series of transactions that combine work spanning its producers and consumers into atomic units.
- It defines a serial order for the messages it consumes and the messages it produces.
- It retains messages it consumes until they have been acknowledged.
- It serializes execution of message listeners registered with its message consumers.

A session can create and service multiple message producers and consumers.

One typical use is to have a thread block on a synchronous **MessageConsumer** (p. 2674) until a message arrives. The thread may then use one or more of the Session's MessageProducers.

If a client desires to have one thread produce messages while others consume them, the client should use a separate session for its producing thread.

Certain rules apply to a session's `close` method and are detailed below.

- There is no need to close the producers and consumers of a closed session.
- The close call will block until a receive call or message listener in progress has completed. A blocked message consumer receive call returns null when this session is closed.
- Closing a transacted session must roll back the transaction in progress.
- The close method is the only **Session** (p. 3460) method that can be called concurrently.
- Invoking any other **Session** (p. 3460) method on a closed session must throw an **IllegalStateException** (p. 2059). Closing a closed session must not throw any exceptions.

### Transacted Sessions

When a **Session** (p. 3460) is created it can be set to operate in a Transaction based mode. Each **Session** (p. 3460) then operates in a single transaction for all Producers and Consumers of that **Session** (p. 3460). Messages sent and received within a Transaction are grouped into an atomic unit that is committed or rolled back together.

For a **MessageProducer** (p. 2809) this implies that all messages sent by the producer are not sent to the Provider until the commit call is made. Rolling back the Transaction results in all produced Messages being dropped.

For a **MessageConsumer** (p. 2674) this implies that all received messages are not Acknowledged until the Commit call is made. Rolling back the Transaction results in all Consumed **Message** (p. 2614) being redelivered to the client, the Provider may allow configuration that limits the Maximum number of redeliveries for a **Message** (p. 2614).

### Since

1.0

## 6.714.2 Member Enumeration Documentation

### 6.714.2.1 enum cms::Session::AcknowledgeMode

#### Enumerator:

**AUTO\_ACKNOWLEDGE** With this acknowledgment mode, the session automatically acknowledges a client's receipt of a message either when the session has successfully returned from a call to receive or when the message listener the session has called to process the message successfully returns.

**DUPS\_OK\_ACKNOWLEDGE** With this acknowledgment mode, the session automatically acknowledges a client's receipt of a message either when the session has successfully returned from a call to receive or when the message listener the session has called to process the message successfully returns. Acknowledgments may be delayed in this mode to increase performance at the cost of the message being redelivered this client fails.

**CLIENT\_ACKNOWLEDGE** With this acknowledgment mode, the client acknowledges a consumed message by calling the message's acknowledge method.

**SESSION\_TRANSACTED** Messages will be consumed when the transaction commits.

**INDIVIDUAL\_ACKNOWLEDGE** **Message** (p. 2614) will be acknowledged individually. Normally the acks sent acknowledge the given message and all messages received before it, this mode only acknowledges one message.

### 6.714.3 Constructor & Destructor Documentation

6.714.3.1 `virtual cms::Session::~~Session ( ) [inline, virtual]`

### 6.714.4 Member Function Documentation

6.714.4.1 `virtual void cms::Session::close ( ) throw ( CMSEException ) [pure virtual]`

Closes this session as well as any active child consumers or producers.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- If an internal error occurs.
--	--------------------------------

Implements `cms::Closeable` (p. 1180).

Implemented in `activemq::cmsutil::PooledSession` (p. 3045), and `activemq::core::ActiveMQSession` (p. 518).

6.714.4.2 `virtual void cms::Session::commit ( ) throw ( CMSEException ) [pure virtual]`

Commits all messages done in this transaction and releases any locks currently held.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- If an internal error occurs.
<b><i>IllegalStateException</i></b> (p. 2059)	- if the method is not called by a transacted session.

Implemented in `activemq::cmsutil::PooledSession` (p. 3045), and `activemq::core::ActiveMQSession` (p. 518).

Referenced by `activemq::cmsutil::PooledSession::commit()`.

6.714.4.3 `virtual QueueBrowser* cms::Session::createBrowser ( const cms::Queue * queue ) throw ( CMSEException ) [pure virtual]`

Creates a new `QueueBrowser` (p. 3243) to peek at Messages on the given `Queue` (p. 3238).

#### Parameters

<i>queue</i>	the <code>Queue</code> (p. 3238) to browse
--------------	--

**Returns**

New **QueueBrowser** (p. 3243) that is owned by the caller.

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- If an internal error occurs.
<b><i>InvalidDestinationException</i></b> (p. 2200)	- if the destination given is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 3046), and **activemq::core::ActiveMQSession** (p. 518).

**6.714.4.4** `virtual QueueBrowser* cms::Session::createBrowser ( const cms::Queue *  
queue, const std::string & selector ) throw ( CMSEException ) [pure  
virtual]`

Creates a new **QueueBrowser** (p. 3243) to peek at Messages on the given **Queue** (p. 3238).

**Parameters**

<i>queue</i>	the <b>Queue</b> (p. 3238) to browse
<i>selector</i>	the <b>Message</b> (p. 2614) selector to filter which messages are browsed.

**Returns**

New **QueueBrowser** (p. 3243) that is owned by the caller.

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- If an internal error occurs.
<b><i>InvalidDestinationException</i></b> (p. 2200)	- if the destination given is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 3045), and **activemq::core::ActiveMQSession** (p. 519).

**6.714.4.5** `virtual BytesMessage* cms::Session::createBytesMessage ( ) throw ( CMSEException ) [pure virtual]`

Creates a **BytesMessage** (p. 1079).

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- If an internal error occurs.
--	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 3046), and **activemq::core::ActiveMQSession** (p. 519).

**6.714.4.6** `virtual BytesMessage* cms::Session::createBytesMessage ( const unsigned char * bytes, int bytesSize ) throw ( CMSEException ) [pure virtual]`

Creates a **BytesMessage** (p. 1079) and sets the payload to the passed value.

#### Parameters

<i>bytes</i>	an array of bytes to set in the message
<i>bytesSize</i>	the size of the bytes array, or number of bytes to use

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- If an internal error occurs.
--	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 3047), and **activemq::core::ActiveMQSession** (p. 520).

**6.714.4.7** `virtual MessageConsumer* cms::Session::createConsumer ( const Destination * destination ) throw ( CMSEException ) [pure virtual]`

Creates a **MessageConsumer** (p. 2674) for the specified destination.

#### Parameters

<i>destination</i>	the <b>Destination</b> (p. 1776) that this consumer receiving messages for.
--------------------	---

#### Returns

pointer to a new **MessageConsumer** (p. 2674) that is owned by the caller ( caller deletes )

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- If an internal error occurs.
<b><i>InvalidDestinationException</i></b> (p. 2200)	- if an invalid destination is specified.

Implemented in **activemq::cmsutil::PooledSession** (p. 3048), and **activemq::core::ActiveMQSession** (p. 521).

**6.714.4.8** `virtual MessageConsumer* cms::Session::createConsumer ( const Destination  
* destination, const std::string & selector ) throw ( CMSEException ) [pure  
virtual]`

Creates a **MessageConsumer** (p. 2674) for the specified destination, using a message selector.

#### Parameters

<i>destination</i>	the <b>Destination</b> (p. 1776) that this consumer receiving messages for.
<i>selector</i>	the <b>Message</b> (p. 2614) Selector to use

#### Returns

pointer to a new **MessageConsumer** (p. 2674) that is owned by the caller ( caller deletes )

#### Exceptions

<b>CMSEException</b> (p. 1190)	- If an internal error occurs.
<b>InvalidDestinationException</b> (p. 2200)	- if an invalid destination is specified.
<b>InvalidSelectorException</b> (p. 2206)	- if the message selector is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 3048), and **activemq::core::ActiveMQSession** (p. 520).

**6.714.4.9** `virtual MessageConsumer* cms::Session::createConsumer ( const Destination  
* destination, const std::string & selector, bool noLocal ) throw ( CMSEException )  
[pure virtual]`

Creates a **MessageConsumer** (p. 2674) for the specified destination, using a message selector.

#### Parameters

<i>destination</i>	the <b>Destination</b> (p. 1776) that this consumer receiving messages for.
<i>selector</i>	the <b>Message</b> (p. 2614) Selector to use
<i>noLocal</i>	if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

#### Returns

pointer to a new **MessageConsumer** (p. 2674) that is owned by the caller ( caller deletes )



**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	- If an internal error occurs.
<b><i>InvalidDestinationException</i></b> (p. 2200)	- if an invalid destination is specified.
<b><i>InvalidSelectorException</i></b> (p. 2206)	- if the message selector is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 3049), and **activemq::core::ActiveMQSession** (p. 520).

**6.714.4.10** `virtual MessageConsumer* cms::Session::createDurableConsumer ( const Topic * destination, const std::string & name, const std::string & selector, bool noLocal = false ) throw ( CMSException ) [pure virtual]`

Creates a durable subscriber to the specified topic, using a **Message** (p. 2614) selector.

Sessions that create durable consumers must use the same client Id as was used the last time the subscription was created in order to receive all messages that were delivered while the client was offline.

**Parameters**

<i>destination</i>	the topic to subscribe to
<i>name</i>	The name used to identify the subscription
<i>selector</i>	the <b>Message</b> (p. 2614) Selector to use
<i>noLocal</i>	if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

**Returns**

pointer to a new durable **MessageConsumer** (p. 2674) that is owned by the caller ( caller deletes )

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	- If an internal error occurs.
<b><i>InvalidDestinationException</i></b> (p. 2200)	- if an invalid destination is specified.
<b><i>InvalidSelectorException</i></b> (p. 2206)	- if the message selector is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 3049), and **activemq::core::ActiveMQSession** (p. 521).

**6.714.4.11** `virtual MapMessage* cms::Session::createMapMessage ( ) throw ( CMSException )` [pure virtual]

Creates a new **MapMessage** (p. 2551).

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- If an internal error occurs.
---	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 3050), and **activemq::core::ActiveMQSession** (p. 522).

**6.714.4.12** `virtual Message* cms::Session::createMessage ( ) throw ( CMSException )` [pure virtual]

Creates a new **Message** (p. 2614).

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- If an internal error occurs.
---	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 3050), and **activemq::core::ActiveMQSession** (p. 522).

**6.714.4.13** `virtual MessageProducer* cms::Session::createProducer ( const Destination * destination ) throw ( CMSException )` [pure virtual]

Creates a **MessageProducer** (p. 2809) to send messages to the specified destination.

#### Parameters

<i>destination</i>	the <b>Destination</b> (p. 1776) to send on
--------------------	---

#### Returns

New **MessageProducer** (p. 2809) that is owned by the caller.

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- If an internal error occurs.
<b><i>InvalidDestinationException</i></b> (p. 2200)	- if an invalid destination is specified.

Implemented in **activemq::cmsutil::PooledSession** (p. 3051), and **activemq::core::ActiveMQSession** (p. 522).

**6.714.4.14** `virtual Queue* cms::Session::createQueue ( const std::string & queueName )  
throw ( CMSEException ) [pure virtual]`

Creates a queue identity given a **Queue** (p. 3238) name.

#### Parameters

<i>queueName</i>	the name of the new <b>Queue</b> (p. 3238)
------------------	--

#### Returns

new **Queue** (p. 3238) pointer that is owned by the caller.

#### Exceptions

<i>CMSEException</i> (p. 1190)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 3051), and **activemq::core::ActiveMQSession** (p. 522).

**6.714.4.15** `virtual StreamMessage* cms::Session::createStreamMessage ( ) throw ( CMSEException ) [pure virtual]`

Creates a new **StreamMessage** (p. 3760).

#### Exceptions

<i>CMSEException</i> (p. 1190)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 3052), and **activemq::core::ActiveMQSession** (p. 523).

**6.714.4.16** `virtual TemporaryQueue* cms::Session::createTemporaryQueue ( ) throw ( CMSEException ) [pure virtual]`

Creates a **TemporaryQueue** (p. 3871) object.

#### Returns

new **TemporaryQueue** (p. 3871) pointer that is owned by the caller.

#### Exceptions

<i>CMSEException</i> (p. 1190)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 3052), and **activemq::core::ActiveMQSession** (p. 523).

**6.714.4.17** `virtual TemporaryTopic* cms::Session::createTemporaryTopic ( ) throw ( CMSEException ) [pure virtual]`

Creates a **TemporaryTopic** (p. 3873) object.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- If an internal error occurs.
--	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 3052), and **activemq::core::ActiveMQSession** (p. 523).

**6.714.4.18** `virtual TextMessage* cms::Session::createTextMessage ( const std::string & text ) throw ( CMSEException ) [pure virtual]`

Creates a new **TextMessage** (p. 3874) and set the text to the value given.

#### Parameters

<i>text</i>	the initial text for the message
-------------	----------------------------------

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- If an internal error occurs.
--	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 3052), and **activemq::core::ActiveMQSession** (p. 524).

**6.714.4.19** `virtual TextMessage* cms::Session::createTextMessage ( ) throw ( CMSEException ) [pure virtual]`

Creates a new **TextMessage** (p. 3874).

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- If an internal error occurs.
--	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 3053), and **activemq::core::ActiveMQSession** (p. 524).

**6.714.4.20** `virtual Topic* cms::Session::createTopic ( const std::string & topicName ) throw ( CMSEException )` [pure virtual]

Creates a topic identity given a **Queue** (p. 3238) name.

#### Parameters

<i>topicName</i>	the name of the new <b>Topic</b> (p. 3930)
------------------	--

#### Returns

new **Topic** (p. 3930) pointer that is owned by the caller.

#### Exceptions

<b>CMSEException</b> (p. 1190)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 3053), and **activemq::core::ActiveMQSession** (p. 524).

**6.714.4.21** `virtual AcknowledgeMode cms::Session::getAcknowledgeMode ( ) const throw ( CMSEException )` [pure virtual]

Returns the acknowledgment mode of the session.

#### Returns

the Sessions Acknowledge Mode

#### Exceptions

<b>CMSEException</b> (p. 1190)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 3053), and **activemq::core::ActiveMQSession** (p. 525).

**6.714.4.22** `virtual bool cms::Session::isTransacted ( ) const throw ( CMSEException )` [pure virtual]

Gets if the Sessions is a Transacted **Session** (p. 3460).

#### Returns

transacted true - false.

#### Exceptions

<b>CMSEException</b> (p. 1190)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 3054), and **activemq::core::ActiveMQSession** (p. 528).

**6.714.4.23** `virtual void cms::Session::recover ( ) throw ( CMSException )` [pure virtual]

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.

All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "re-delivered"
- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to stop and restart message delivery due to some internal error.
<b><i>IllegalStateException</i></b> (p. 2059)	- if the method is called by a transacted session.

Implemented in **activemq::cmsutil::PooledSession** (p. 3054), and **activemq::core::ActiveMQSession** (p. 528).

**6.714.4.24** `virtual void cms::Session::rollback ( ) throw ( CMSException )` [pure virtual]

Rolls back all messages done in this transaction and releases any locks currently held.

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- If an internal error occurs.
<b><i>IllegalStateException</i></b> (p. 2059)	- if the method is not called by a transacted session.

Implemented in **activemq::cmsutil::PooledSession** (p. 3055), and **activemq::core::ActiveMQSession** (p. 530).

**6.714.4.25** virtual void cms::Session::unsubscribe ( const std::string & *name* ) throw ( CMSException ) [pure virtual]

Unsubscribes a durable subscription that has been created by a client.

This method deletes the state being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active **MessageConsumer** (p. 2674) or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

#### Parameters

<i>name</i>	The name used to identify this subscription
-------------	---

#### Exceptions

<b>CMSException</b> (p. 1190)	- If an internal error occurs.
----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 3055), and **activemq::core::ActiveMQSession** (p. 531).

The documentation for this class was generated from the following file:

- src/main/cms/Session.h

## 6.715 activemq::cmsutil::SessionCallback Class Reference

Callback for executing any number of operations on a provided CMS Session.

```
#include <src/main/activemq/cmsutil/SessionCallback.h>
```

Inheritance diagram for activemq::cmsutil::SessionCallback:

#### Public Member Functions

- virtual ~SessionCallback ()
- virtual void doInCms (cms::Session \*session)=0 throw ( cms::CMSException )

*Execute any number of operations against the supplied CMS session.*

### 6.715.1 Detailed Description

Callback for executing any number of operations on a provided CMS Session.

## 6.715.2 Constructor & Destructor Documentation

6.715.2.1 `virtual activemq::cmsutil::SessionCallback::~SessionCallback ( ) [inline, virtual]`

## 6.715.3 Member Function Documentation

6.715.3.1 `virtual void activemq::cmsutil::SessionCallback::doInCms ( cms::Session * session ) throw ( cms::CMSException ) [pure virtual]`

Execute any number of operations against the supplied CMS session.

### Parameters

<i>session</i>	the CMS Session
----------------	-----------------

### Exceptions

<i>cms::CMSException</i> (p. 1190)	if thrown by CMS API methods
---------------------------------------	------------------------------

Implemented in `activemq::cmsutil::CmsTemplate::ProducerExecutor` (p. 3156), and `activemq::cmsutil::CmsTemplate::ReceiveExecutor` (p. 3266).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/SessionCallback.h`

## 6.716 activemq::commands::SessionId Class Reference

```
#include <src/main/activemq/commands/SessionId.h>
```

Inheritance diagram for `activemq::commands::SessionId`:

### Public Types

- `typedef decaf::lang::PointerComparator< SessionId > COMPARATOR`

### Public Member Functions

- `SessionId ()`
- `SessionId (const SessionId &other)`
- `SessionId (const ConnectionId *connectionId, long long sessionId)`
- `SessionId (const ProducerId *producerId)`
- `SessionId (const ConsumerId *consumerId)`



- virtual `~SessionId ()`
- virtual unsigned char `getDataStructureType ()` const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual `SessionId * cloneDataStructure ()` const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void `copyDataStructure (const DataStructure *src)`  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string `toString ()` const  
*Returns a string containing the information for this DataStructure (p. 1713) such as its type and value of its elements.*
- virtual bool `equals (const DataStructure *value)` const  
*Compares the DataStructure (p. 1713) passed in to this one, and returns if they are equivalent.*
- const `Pointer< ConnectionId > & getParentId ()` const
- virtual const std::string & `getConnectionId ()` const
- virtual std::string & `getConnectionId ()`
- virtual void `setConnectionId (const std::string &connectionId)`
- virtual long long `getValue ()` const
- virtual void `setValue (long long value)`
- virtual int `compareTo (const SessionId &value)` const
- virtual bool `equals (const SessionId &value)` const
- virtual bool `operator== (const SessionId &value)` const
- virtual bool `operator< (const SessionId &value)` const
- `SessionId & operator= (const SessionId &other)`

### Static Public Attributes

- static const unsigned char `ID_SESSIONID = 121`

### Protected Attributes

- std::string `connectionId`
- long long `value`

**6.716.1 Member Typedef Documentation**

6.716.1.1 `typedef decaf::lang::PointerComparator<SessionId>`  
`activemq::commands::SessionId::COMPARATOR`

**6.716.2 Constructor & Destructor Documentation**

6.716.2.1 `activemq::commands::SessionId::SessionId ( )`

6.716.2.2 `activemq::commands::SessionId::SessionId ( const SessionId & other )`

6.716.2.3 `activemq::commands::SessionId::SessionId ( const ConnectionId * connectionId,`  
`long long sessionId )`

6.716.2.4 `activemq::commands::SessionId::SessionId ( const ProducerId * producerId )`

6.716.2.5 `activemq::commands::SessionId::SessionId ( const ConsumerId * consumerId )`

6.716.2.6 `virtual activemq::commands::SessionId::~~SessionId ( ) [virtual]`

**6.716.3 Member Function Documentation**

6.716.3.1 `virtual SessionId* activemq::commands::SessionId::cloneDataStructure ( ) const`  
`[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

**Returns**

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1714).

6.716.3.2 `virtual int activemq::commands::SessionId::compareTo ( const SessionId & value )`  
`const [virtual]`

6.716.3.3 `virtual void activemq::commands::SessionId::copyDataStructure ( const`  
`DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

**Parameters**

<code>src</code>	- Source Object
------------------	-----------------

Implements `activemq::commands::DataStructure` (p. 1715).

**6.716.3.4** `virtual bool activemq::commands::SessionId::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1716).

**6.716.3.5** `virtual bool activemq::commands::SessionId::equals ( const SessionId & value ) const [virtual]`

**6.716.3.6** `virtual std::string& activemq::commands::SessionId::getConnectionId ( ) [virtual]`

**6.716.3.7** `virtual const std::string& activemq::commands::SessionId::getConnectionId ( ) const [virtual]`

**6.716.3.8** `virtual unsigned char activemq::commands::SessionId::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

### Returns

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

- 6.716.3.9 `const Pointer<ConnectionId>& activemq::commands::SessionId::getParentId ( ) const`
- 6.716.3.10 `virtual long long activemq::commands::SessionId::getValue ( ) const`  
[virtual]
- 6.716.3.11 `virtual bool activemq::commands::SessionId::operator< ( const SessionId & value ) const` [virtual]
- 6.716.3.12 `SessionId& activemq::commands::SessionId::operator= ( const SessionId & other )`
- 6.716.3.13 `virtual bool activemq::commands::SessionId::operator==( const SessionId & value ) const` [virtual]
- 6.716.3.14 `virtual void activemq::commands::SessionId::setConnectionId ( const std::string & connectionId )` [virtual]
- 6.716.3.15 `virtual void activemq::commands::SessionId::setValue ( long long value )`  
[virtual]
- 6.716.3.16 `virtual std::string activemq::commands::SessionId::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 841).

### 6.716.4 Field Documentation

- 6.716.4.1 `std::string activemq::commands::SessionId::connectionId`  
[protected]
- 6.716.4.2 `const unsigned char activemq::commands::SessionId::ID_SESSIONID = 121` [static]

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

- 6.716.4.3 `long long activemq::commands::SessionId::value` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/SessionId.h`

## 6.717 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3481).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/SessionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller:

### Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.717.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3481). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.717.2 Constructor & Destructor Documentation

6.717.2.1 **activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::SessionIdMarshaller**  
( ) [inline]

6.717.2.2 **virtual activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::~~SessionIdMarshaller**  
( ) [inline, virtual]

### 6.717.3 Member Function Documentation

6.717.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.717.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.717.3.3 **virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

## 6.717 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller Class Reference 3487

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.717.3.4** virtual void **activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.717.3.5** virtual int **activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::tightMarshal1** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.717.3.6  virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.717.3.7  virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:



- src/main/activemq/wireformat/openwire/marshal/v2/SessionIdMarshaller.h

## 6.718 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3485).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/SessionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller:

### Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.718.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3485). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.718.2 Constructor & Destructor Documentation

6.718.2.1 **activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::SessionIdMarshaller**  
( ) [inline]

6.718.2.2 **virtual activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::~~SessionIdMarshaller**  
( ) [inline, virtual]

### 6.718.3 Member Function Documentation

6.718.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.718.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.718.3.3 **virtual void activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

## 6.718 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller Class Reference 3491

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

```
6.718.3.4 virtual void activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::looseUnmarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
  * dataStructure, decaf::io::DataInputStream * dataIn ) throw (  
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

```
6.718.3.5 virtual int activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::tightMarshal1  
( OpenWireFormat * wireFormat, commands::DataStructure *  
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )  
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.718.3.6  virtual void activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.718.3.7  virtual void activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/SessionIdMarshaller.h

## 6.719 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3489).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/SessionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller:

### Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.719.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3489). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.719.2 Constructor & Destructor Documentation

6.719.2.1 **activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::SessionIdMarshaller**  
( ) [inline]

6.719.2.2 **virtual activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::~~SessionIdMarshaller**  
( ) [inline, virtual]

### 6.719.3 Member Function Documentation

6.719.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.719.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.719.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

## 6.719 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller Class Reference 3495

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

```
6.719.3.4 virtual void activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::looseUnmarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
  * dataStructure, decaf::io::DataInputStream * dataIn ) throw (  
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

```
6.719.3.5 virtual int activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::tightMarshal1  
( OpenWireFormat * wireFormat, commands::DataStructure *  
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )  
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.719.3.6  virtual void activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.719.3.7  virtual void activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:



- src/main/activemq/wireformat/openwire/marshal/v6/SessionIdMarshaller.h

## 6.720 activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3493).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/SessionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller:

### Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**)  
throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.720.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3493). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.720.2 Constructor & Destructor Documentation

6.720.2.1 **activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::SessionIdMarshaller**  
( ) [inline]

6.720.2.2 **virtual activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::~~SessionIdMarshaller**  
( ) [inline, virtual]

### 6.720.3 Member Function Documentation

6.720.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.720.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.720.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

## 6.720 activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller Class Reference 3499

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

```
6.720.3.4 virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::looseUnmarshal  
( OpenWireFormat * wireFormat, commands::DataStructure  
  * dataStructure, decaf::io::DataInputStream * dataIn ) throw (  
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

```
6.720.3.5 virtual int activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::tightMarshal1  
( OpenWireFormat * wireFormat, commands::DataStructure *  
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )  
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.720.3.6  virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.720.3.7  virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/SessionIdMarshaller.h

## 6.721 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3497).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/SessionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller:

### Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.721.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3497). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.721.2 Constructor & Destructor Documentation

6.721.2.1 **activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::SessionIdMarshaller**  
( ) [inline]

6.721.2.2 **virtual activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::~~SessionIdMarshaller**  
( ) [inline, virtual]

### 6.721.3 Member Function Documentation

6.721.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.721.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.721.3.3 **virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

## 6.721 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller Class Reference 3503

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.721.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.721.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.721.3.6  virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.721.3.7  virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:



- src/main/activemq/wireformat/openwire/marshal/v3/SessionIdMarshaller.h

## 6.722 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3501).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/SessionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller:

### Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.722.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3501). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.722.2 Constructor & Destructor Documentation

6.722.2.1 **activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::SessionIdMarshaller**  
( ) [inline]

6.722.2.2 **virtual activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::~~SessionIdMarshaller**  
( ) [inline, virtual]

### 6.722.3 Member Function Documentation

6.722.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.722.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.722.3.3 **virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.722.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.722.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.722.3.6  virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.722.3.7  virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshall/v1/SessionIdMarshaller.h

## 6.723 activemq::commands::SessionInfo Class Reference

```
#include <src/main/activemq/commands/SessionInfo.h>
```

Inheritance diagram for activemq::commands::SessionInfo:

### Public Member Functions

- **SessionInfo** ()
- virtual **~SessionInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **SessionInfo** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- unsigned int **getAckMode** () const
- void **setAckMode** (unsigned int mode)
- **Pointer**< **RemoveInfo** > **createRemoveCommand** () const
- virtual const **Pointer**< **SessionId** > & **getSessionId** () const
- virtual **Pointer**< **SessionId** > & **getSessionId** ()
- virtual void **setSessionId** (const **Pointer**< **SessionId** > &sessionId)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

**Static Public Attributes**

- static const unsigned char **ID\_SESSIONINFO** = 4

**Protected Attributes**

- **Pointer< SessionId > sessionId**

**6.723.1 Constructor & Destructor Documentation**

6.723.1.1 **activemq::commands::SessionInfo::SessionInfo ( )**

6.723.1.2 **virtual activemq::commands::SessionInfo::~~SessionInfo ( )** [virtual]

**6.723.2 Member Function Documentation**

6.723.2.1 **virtual SessionInfo\* activemq::commands::SessionInfo::cloneDataStructure ( )**  
**const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

**Returns**

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

6.723.2.2 **virtual void activemq::commands::SessionInfo::copyDataStructure ( const DataStructure \* *src* )** [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

**Parameters**

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

**6.723.2.3** `Pointer<RemoveInfo> activemq::commands::SessionInfo::createRemoveCommand ( ) const`

**6.723.2.4** `virtual bool activemq::commands::SessionInfo::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

**6.723.2.5** `unsigned int activemq::commands::SessionInfo::getAckMode ( ) const [inline]`

**6.723.2.6** `virtual unsigned char activemq::commands::SessionInfo::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1713) type copy.

Implements **activemq::commands::DataStructure** (p. 1717).

**6.723.2.7** `virtual const Pointer<SessionId>& activemq::commands::SessionInfo::getSessionId ( ) const [virtual]`

**6.723.2.8** `virtual Pointer<SessionId>& activemq::commands::SessionInfo::getSessionId ( ) [virtual]`

**6.723.2.9** `void activemq::commands::SessionInfo::setAckMode ( unsigned int mode ) [inline]`

**6.723.2.10** `virtual void activemq::commands::SessionInfo::setSessionId ( const Pointer<SessionId> & sessionId ) [virtual]`

**6.723.2.11** `virtual std::string activemq::commands::SessionInfo::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

**Returns**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

**6.723.2.12** `virtual Pointer<Command> activemq::commands::SessionInfo::visit  
( activemq::state::CommandVisitor * visitor ) throw (  
exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

**Returns**

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

**6.723.3 Field Documentation**

**6.723.3.1** `const unsigned char activemq::commands::SessionInfo::ID_ -  
SESSIONINFO = 4 [static]`

**6.723.3.2** `Pointer<SessionId> activemq::commands::SessionInfo::sessionId  
[protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/SessionInfo.h`

## **6.724 activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller Class Reference**

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3508).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/SessionInfoMars
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller**:

**Public Member Functions**

- **SessionInfoMarshaller ()**



- virtual `~SessionInfoMarshaller ()`
- virtual `commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshall an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.724.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3508). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.724.2 Constructor & Destructor Documentation

6.724.2.1 `activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::SessionInfoMarshaller ( ) [inline]`

6.724.2.2 `virtual activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::~~SessionInfoMarshaller ( ) [inline, virtual]`

## 6.724.3 Member Function Documentation

6.724.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.724.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.724.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 801).

**6.724.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 802).

**6.724.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 803).

```
6.724.3.6 virtual void activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 804).

```
6.724.3.7 virtual void activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 806).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/SessionInfoMarshaller.h

## 6.725 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3513).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller:

### Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.725.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3513). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.725.2 Constructor & Destructor Documentation

6.725.2.1 **activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::SessionInfoMarshaller**  
( ) [inline]

6.725.2.2 **virtual activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::~~SessionInfoMarshaller**  
( ) [inline, virtual]

### 6.725.3 Member Function Documentation

6.725.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.725.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.725.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 794).

**6.725.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 795).

**6.725.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 796).

```
6.725.3.6 virtual void activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 797).

```
6.725.3.7 virtual void activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.



## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h

## 6.726 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3517).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/SessionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller**:

## Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.726.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3517). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.726.2 Constructor & Destructor Documentation

6.726.2.1 **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::SessionInfoMarshaller**  
 ( ) [inline]

6.726.2.2 **virtual activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::~~SessionInfoMarshaller**  
 ( ) [inline, virtual]

### 6.726.3 Member Function Documentation

6.726.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::createObject** ( )  
 const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.726.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.726.3.3** virtual void **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::looseMarshal**  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
 \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw (  
**decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**  
 (p. 787).

**6.726.3.4** virtual void **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::looseUnmarshal**  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
 \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw (  
**decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**  
 (p. 788).

```

6.726.3.5  virtual int activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 789).

```

6.726.3.6  virtual void activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 790).

6.726.3.7 virtual void activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 792).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**SessionInfoMarshaller.h**

## 6.727 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3521).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller:

#### Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.727.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3521). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.727.2 Constructor & Destructor Documentation

6.727.2.1 **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::SessionInfoMarshaller**  
( ) [inline]

6.727.2.2 **virtual activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::~~SessionInfoMarshaller**  
( ) [inline, virtual]

### 6.727.3 Member Function Documentation

6.727.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.727.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.727.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 772).

**6.727.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 774).

```
6.727.3.5  virtual int activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 775).

```
6.727.3.6  virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled



## 6.728 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller

### Class Reference

3529

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 776).

**6.727.3.7** `virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h

## 6.728 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3525).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/SessionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller**:

## Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.728.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3525). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.728.2 Constructor & Destructor Documentation

6.728.2.1 `activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::SessionInfoMarshaller ( ) [inline]`

6.728.2.2 `virtual activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::~~SessionInfoMarshaller ( ) [inline, virtual]`

## 6.728.3 Member Function Documentation

6.728.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.728.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.728.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 808).

**6.728.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 809).

**6.728.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 810).

**6.728.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 811).

**6.728.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 813).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/SessionInfoMarshaller.h`

## 6.729 `activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3530).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/SessionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller`:

### Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.729.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3530). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.729.2 Constructor & Destructor Documentation

- 6.729.2.1 **activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::SessionInfoMarshaller**  
( ) [inline]
- 6.729.2.2 **virtual activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::~~SessionInfoMarshaller**  
( ) [inline, virtual]

### 6.729.3 Member Function Documentation

- 6.729.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

- 6.729.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

- 6.729.3.3 **virtual void activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 780).

**6.729.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 781).

**6.729.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 782).

**6.729.3.6** virtual void activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::tightMarshal2  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure  
 \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
 utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
 [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 783).

**6.729.3.7** virtual void activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::tightUnmarshal  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream  
 \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**SessionInfoMarshaller.h**

**6.730 activemq::cmsutil::SessionPool Class Reference**

A pool of CMS sessions from the same connection and with the same acknowledge mode.

```
#include <src/main/activemq/cmsutil/SessionPool.h>
```

**Public Member Functions**

- **SessionPool** (**cms::Connection** \*connection, **cms::Session::AcknowledgeMode** ackMode, **ResourceLifecycleManager** \*resourceLifecycleManager)  
*Constructs a session pool.*
- virtual **~SessionPool** ()  
*Destroys the pooled session objects, but not the underlying session resources.*
- virtual **PooledSession** \* **takeSession** () throw ( **cms::CMSEException** )  
*Takes a session from the pool, creating one if necessary.*
- virtual void **returnSession** (**PooledSession** \*session)  
*Returns a session to the pool.*
- **ResourceLifecycleManager** \* **getResourceLifecycleManager** ()

**Protected Member Functions**

- **SessionPool** (const **SessionPool** &)
- **SessionPool** & **operator=** (const **SessionPool** &)

**6.730.1 Detailed Description**

A pool of CMS sessions from the same connection and with the same acknowledge mode. Internal session resources are managed through a provided **ResourceLifecycleManager** (p. 3376), not by this pool. This class is thread-safe.

## 6.730.2 Constructor & Destructor Documentation

**6.730.2.1** `activemq::cmsutil::SessionPool::SessionPool ( const SessionPool & )`  
`[inline, protected]`

**6.730.2.2** `activemq::cmsutil::SessionPool::SessionPool ( cms::Connection * connection, cms::Session::AcknowledgeMode ackMode, ResourceLifecycleManager * resourceLifecycleManager )`

Constructs a session pool.

### Parameters

<i>connection</i>	the connection to be used for creating all sessions.
<i>ackMode</i>	the acknowledge mode to be used for all sessions
<i>resourceLifecycleManager</i>	the object responsible for managing the lifecycle of any allocated <b>cms::Session</b> (p. 3460) resources.

**6.730.2.3** `virtual activemq::cmsutil::SessionPool::~SessionPool ( )` `[virtual]`

Destroys the pooled session objects, but not the underlying session resources.

That is the job of the **ResourceLifecycleManager** (p. 3376).

## 6.730.3 Member Function Documentation

**6.730.3.1** `ResourceLifecycleManager* activemq::cmsutil::SessionPool::getResourceLifecycleManager ( )`  
`[inline]`

**6.730.3.2** `SessionPool& activemq::cmsutil::SessionPool::operator= ( const SessionPool & )`  
`[inline, protected]`

**6.730.3.3** `virtual void activemq::cmsutil::SessionPool::returnSession ( PooledSession * session )` `[virtual]`

Returns a session to the pool.

### Parameters

<i>session</i>	the session to be returned.
----------------	-----------------------------

**6.730.3.4** `virtual PooledSession* activemq::cmsutil::SessionPool::takeSession ( ) throw ( cms::CMSException ) [virtual]`

Takes a session from the pool, creating one if necessary.

#### Returns

the pooled session object

#### Exceptions

<i>cms::CMSException</i> (p. 1190)	if an error occurred
---------------------------------------	----------------------

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/SessionPool.h

## 6.731 activemq::state::SessionState Class Reference

```
#include <src/main/activemq/state/SessionState.h>
```

#### Public Member Functions

- **SessionState** (const **Pointer**< **SessionInfo** > &info)
- virtual **~SessionState** ()
- std::string **toString** () const
- const **Pointer**< **SessionInfo** > **getInfo** () const
- void **addProducer** (const **Pointer**< **ProducerInfo** > &info)
- **Pointer**< **ProducerState** > **removeProducer** (const **Pointer**< **ProducerId** > &id)
- void **addConsumer** (const **Pointer**< **ConsumerInfo** > &info)
- **Pointer**< **ConsumerState** > **removeConsumer** (const **Pointer**< **ConsumerId** > &id)
- std::vector< **Pointer**< **ProducerState** > > **getProducerStates** () const
- **Pointer**< **ProducerState** > **getProducerState** (const **Pointer**< **ProducerId** > &id)
- std::vector< **Pointer**< **ConsumerState** > > **getConsumerStates** () const
- **Pointer**< **ConsumerState** > **getConsumerState** (const **Pointer**< **ConsumerId** > &id)
- void **checkShutdown** () const
- void **shutdown** ()

### 6.731.1 Constructor & Destructor Documentation

6.731.1.1 `activemq::state::SessionState::SessionState ( const Pointer< SessionInfo > & info )`

6.731.1.2 `virtual activemq::state::SessionState::~~SessionState ( ) [virtual]`

### 6.731.2 Member Function Documentation

6.731.2.1 `void activemq::state::SessionState::addConsumer ( const Pointer< ConsumerInfo > & info ) [inline]`

6.731.2.2 `void activemq::state::SessionState::addProducer ( const Pointer< ProducerInfo > & info )`

6.731.2.3 `void activemq::state::SessionState::checkShutdown ( ) const`

6.731.2.4 `Pointer<ConsumerState> activemq::state::SessionState::getConsumerState ( const Pointer< ConsumerId > & id ) [inline]`

6.731.2.5 `std::vector< Pointer<ConsumerState> > activemq::state::SessionState::getConsumerStates ( ) const [inline]`

6.731.2.6 `const Pointer<SessionInfo> activemq::state::SessionState::getInfo ( ) const [inline]`

6.731.2.7 `Pointer<ProducerState> activemq::state::SessionState::getProducerState ( const Pointer< ProducerId > & id ) [inline]`

6.731.2.8 `std::vector< Pointer<ProducerState> > activemq::state::SessionState::getProducerStates ( ) const [inline]`

6.731.2.9 `Pointer<ConsumerState> activemq::state::SessionState::removeConsumer ( const Pointer< ConsumerId > & id ) [inline]`

6.731.2.10 `Pointer<ProducerState> activemq::state::SessionState::removeProducer ( const Pointer< ProducerId > & id )`

6.731.2.11 `void activemq::state::SessionState::shutdown ( ) [inline]`

6.731.2.12 `std::string activemq::state::SessionState::toString ( ) const`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/SessionState.h`

## 6.732 decaf::util::Set< E > Class Template Reference

A collection that contains no duplicate elements.

```
#include <src/main/decaf/util/Set.h>
```

Inheritance diagram for decaf::util::Set< E >:

### Public Member Functions

- virtual `~Set()`

#### 6.732.1 Detailed Description

```
template<typename E> class decaf::util::Set< E >
```

A collection that contains no duplicate elements. More formally, sets contain no pair of elements `e1` and `e2` such that `e1 == e2`, and at most one null element. As implied by its name, this interface models the mathematical set abstraction.

The additional stipulation on constructors is, not surprisingly, that all constructors must create a set that contains no duplicate elements (as defined above).

Note: Great care must be exercised if mutable objects are used as set elements. The behavior of a set is not specified if the value of an object is changed in a manner that affects equals comparisons while the object is an element in the set.

Since

1.0

#### 6.732.2 Constructor & Destructor Documentation

6.732.2.1 `template<typename E> virtual decaf::util::Set< E >::~~Set( ) [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Set.h`

## 6.733 decaf::lang::Short Class Reference

```
#include <src/main/decaf/lang/Short.h>
```

Inheritance diagram for decaf::lang::Short:

## Public Member Functions

- **Short** (short value)
- **Short** (const std::string &value) throw ( exceptions::NumberFormatException )
- virtual ~**Short** ()
- virtual int **compareTo** (const **Short** &s) const  
*Compares this **Short** (p. 3538) instance with another.*
- bool **equals** (const **Short** &s) const
- virtual bool **operator==** (const **Short** &s) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Short** &s) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual int **compareTo** (const short &s) const  
*Compares this **Short** (p. 3538) instance with another.*
- bool **equals** (const short &s) const
- virtual bool **operator==** (const short &s) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const short &s) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- std::string **toString** () const
- virtual double **doubleValue** () const  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const  
*Answers the float value which the receiver represents.*
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual short **shortValue** () const  
*Answers the short value which the receiver represents.*
- virtual int **intValue** () const  
*Answers the int value which the receiver represents.*
- virtual long long **longValue** () const  
*Answers the long value which the receiver represents.*

### Static Public Member Functions

- static std::string **toString** (short value)
- static **Short decode** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Decodes a **String** (p. 3775) into a **Short** (p. 3538).*
- static short **reverseBytes** (short value)  
*Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified short value.*
- static short **parseShort** (const std::string &s, int radix) throw ( exceptions::NumberFormatException )  
*Parses the string argument as a signed short in the radix specified by the second argument.*
- static short **parseShort** (const std::string &s) throw ( exceptions::NumberFormatException )  
*Parses the string argument as a signed decimal short.*
- static **Short valueOf** (short value)  
*Returns a **Short** (p. 3538) instance representing the specified short value.*
- static **Short valueOf** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Returns a **Short** (p. 3538) object holding the value given by the specified std::string.*
- static **Short valueOf** (const std::string &value, int radix) throw ( exceptions::NumberFormatException )  
*Returns a **Short** (p. 3538) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

### Static Public Attributes

- static const int **SIZE** = 16  
*Size of this objects primitive type in bits.*
- static const short **MAX\_VALUE** = (short)0x7FFF  
*Max Value for this Object's primitive type.*
- static const short **MIN\_VALUE** = (short)0x8000  
*Max Value for this Object's primitive type.*



## 6.733.1 Constructor & Destructor Documentation

### 6.733.1.1 decaf::lang::Short::Short ( short *value* )

#### Parameters

<i>value</i>	- short to wrap
--------------	-----------------

### 6.733.1.2 decaf::lang::Short::Short ( const std::string & *value* ) throw ( exceptions::NumberFormatException )

#### Parameters

<i>value</i>	- string value to convert to short and wrap
--------------	---

#### Exceptions

<i>NumberFormatException</i>	
------------------------------	--

### 6.733.1.3 virtual decaf::lang::Short::~~Short ( ) [inline, virtual]

## 6.733.2 Member Function Documentation

### 6.733.2.1 virtual unsigned char decaf::lang::Short::byteValue ( ) const [inline, virtual]

Answers the byte value which the receiver represents.

#### Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2919).

### 6.733.2.2 virtual int decaf::lang::Short::compareTo ( const short & *s* ) const [virtual]

Compares this **Short** (p. 3538) instance with another.

#### Parameters

<i>s</i>	- the <b>Short</b> (p. 3538) instance to be compared
----------	--

#### Returns

zero if this object represents the same short value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **short** > (p. 1249).

**6.733.2.3** `virtual int decaf::lang::Short::compareTo ( const Short & s ) const` [virtual]

Compares this **Short** (p. 3538) instance with another.

#### Parameters

<i>s</i>	- the <b>Short</b> (p. 3538) instance to be compared
----------	--

#### Returns

zero if this object represents the same short value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **Short** > (p. 1249).

**6.733.2.4** `static Short decaf::lang::Short::decode ( const std::string & value ) throw ( exceptions::NumberFormatException )` [static]

Decodes a **String** (p. 3775) into a **Short** (p. 3538).

Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the **Short.parseShort** (p. 3545) method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a **NumberFormatException** will be thrown. The result is negated if first character of the specified **String** (p. 3775) is the minus sign. No whitespace characters are permitted in the string.

#### Parameters

<i>value</i>	- The string to decode
--------------	------------------------

#### Returns

a **Short** (p. 3538) object containing the decoded value

#### Exceptions

<i>NumberFormatException</i>	if the string is not formatted correctly.
------------------------------	---

**6.733.2.5** `virtual double decaf::lang::Short::doubleValue ( ) const` [inline, virtual]

Answers the double value which the receiver represents.

**Returns**

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2919).

**6.733.2.6** `bool decaf::lang::Short::equals ( const Short & s ) const` `[inline, virtual]`

**Returns**

true if the two **Short** (p. 3538) Objects have the same value.

Implements **decaf::lang::Comparable< Short >** (p. 1250).

**6.733.2.7** `bool decaf::lang::Short::equals ( const short & s ) const` `[inline, virtual]`

**Returns**

true if the two **Short** (p. 3538) Objects have the same value.

Implements **decaf::lang::Comparable< short >** (p. 1250).

**6.733.2.8** `virtual float decaf::lang::Short::floatValue ( ) const` `[inline, virtual]`

Answers the float value which the receiver represents.

**Returns**

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2920).

**6.733.2.9** `virtual int decaf::lang::Short::intValue ( ) const` `[inline, virtual]`

Answers the int value which the receiver represents.

**Returns**

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2920).

**6.733.2.10** `virtual long long decaf::lang::Short::longValue ( ) const` `[inline, virtual]`

Answers the long value which the receiver represents.

**Returns**

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2920).

**6.733.2.11** `virtual bool decaf::lang::Short::operator< ( const Short & s ) const` `[inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

**Parameters**

<code>s</code>	- the value to be compared to this one.
----------------	---

**Returns**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Short >** (p. 1250).

**6.733.2.12** `virtual bool decaf::lang::Short::operator< ( const short & s ) const` `[inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

**Parameters**

<code>s</code>	- the value to be compared to this one.
----------------	---

**Returns**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< short >** (p. 1250).

**6.733.2.13** `virtual bool decaf::lang::Short::operator== ( const Short & s ) const` `[inline, virtual]`

Compares equality between this object and the one passed.

**Parameters**

<code>s</code>	- the value to be compared to this one.
----------------	---

**Returns**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Short** > (p. 1250).

**6.733.2.14** `virtual bool decaf::lang::Short::operator==( const short & s ) const` [inline, virtual]

Compares equality between this object and the one passed.

**Parameters**

<i>s</i>	- the value to be compared to this one.
----------	---

**Returns**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **short** > (p. 1250).

**6.733.2.15** `static short decaf::lang::Short::parseShort ( const std::string & s, int radix ) throw ( exceptions::NumberFormatException )` [static]

Parses the string argument as a signed short in the radix specified by the second argument.

The characters in the string must all be digits, of the specified radix (as determined by whether **Character.digit(char, int)** (p. 1130) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type **NumberFormatException** is thrown if any of the following situations occurs: \* The first argument is null or is a string of length zero. \* The radix is either smaller than **Character.MIN\_RADIX** (p. 1134) or larger than **Character.MAX\_RADIX** (p. 1134). \* Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. \* The value represented by the string is not a value of type short.

**Parameters**

<i>s</i>	- the <b>String</b> (p. 3775) containing the short representation to be parsed
<i>radix</i>	- the radix to be used while parsing s

**Returns**

the short represented by the string argument in the specified radix.

**Exceptions**

<i>NumberFormatException</i>	- If <b>String</b> (p. 3775) does not contain a parsable short.
------------------------------	---

**6.733.2.16** `static short decaf::lang::Short::parseShort ( const std::string & s ) throw ( exceptions::NumberFormatException )` [static]

Parses the string argument as a signed decimal short.

The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting short value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseShort( const std::string, int )` method.

#### Parameters

<i>s</i>	- <b>String</b> (p. 3775) to convert to a short
----------	---

#### Returns

the converted short value

#### Exceptions

<i>NumberFormatException</i>	if the string is not a short.
------------------------------	-------------------------------

**6.733.2.17** `static short decaf::lang::Short::reverseBytes ( short value )` [static]

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified short value.

#### Parameters

<i>value</i>	- the short whose bytes we are to reverse
--------------	---

#### Returns

the reversed short.

**6.733.2.18** `virtual short decaf::lang::Short::shortValue ( ) const` [inline, virtual]

Answers the short value which the receiver represents.

#### Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2920).

**6.733.2.19** `static std::string decaf::lang::Short::toString ( short value )` [static]

#### Returns

a string representing the primitive value as Base 10

**6.733.2.20** `std::string decaf::lang::Short::toString ( ) const`**Returns**

this **Short** (p. 3538) Object as a **String** (p. 3775) Representation

**6.733.2.21** `static Short decaf::lang::Short::valueOf ( short value ) [static]`

Returns a **Short** (p. 3538) instance representing the specified short value.

**Parameters**

<i>value</i>	- the short to wrap
--------------	---------------------

**Returns**

the new **Short** (p. 3538) object wrapping value.

**6.733.2.22** `static Short decaf::lang::Short::valueOf ( const std::string & value ) throw ( exceptions::NumberFormatException ) [static]`

Returns a **Short** (p. 3538) object holding the value given by the specified std::string.

The argument is interpreted as representing a signed decimal short, exactly as if the argument were given to the `parseShort( std::string )` method. The result is a **Short** (p. 3538) object that represents the short value specified by the string.

**Parameters**

<i>value</i>	- std::string to parse as base 10
--------------	-----------------------------------

**Returns**

new **Short** (p. 3538) Object wrapping the primitive

**Exceptions**

<i>NumberFormatException</i>	if the string is not a decimal short.
------------------------------	---------------------------------------

**6.733.2.23** `static Short decaf::lang::Short::valueOf ( const std::string & value, int radix ) throw ( exceptions::NumberFormatException ) [static]`

Returns a **Short** (p. 3538) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed short in the radix specified by the second argument, exactly as if the argument were given to the `parseShort( std::string, int )` method. The result is a **Short** (p. 3538) object that represents the short

value specified by the string.

Parameters

<i>value</i>	- std::string to parse as base ( radix )
<i>radix</i>	- base of the string to parse.

Returns

new **Short** (p. 3538) Object wrapping the primitive

Exceptions

<i>NumberFormatException</i>	if the string is not a valid short.
------------------------------	-------------------------------------

6.733.3 Field Documentation

6.733.3.1 `const short decaf::lang::Short::MAX_VALUE = (short)0x7FFF` [static]

Max Value for this Object's primitive type.

6.733.3.2 `const short decaf::lang::Short::MIN_VALUE = (short)0x8000` [static]

Max Value for this Object's primitive type.

6.733.3.3 `const int decaf::lang::Short::SIZE = 16` [static]

Size of this objects primitive type in bits.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/Short.h

6.734 decaf::internal::nio::ShortArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/ShortArrayBuffer.h>
```

Inheritance diagram for decaf::internal::nio::ShortArrayBuffer:

Public Member Functions

- **ShortArrayBuffer** (int size, bool readOnly=false) throw ( decaf::lang::exceptions::IllegalArgumentExcep )



Creates a **ShortArrayBuffer** (p. 3548) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

- **ShortArrayBuffer** (short \*array, int size, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a **ShortArrayBuffer** (p. 3548) object that wraps the given array.

- **ShortArrayBuffer** (const decaf::lang::Pointer< ByteArrayAdapter > &array, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

- **ShortArrayBuffer** (const ShortArrayBuffer &other)

Create a **ShortArrayBuffer** (p. 3548) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.

- virtual ~**ShortArrayBuffer** ()
- virtual short \* **array** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

Returns the short array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

the array that backs this **Buffer** (p. 936)

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
UnsupportedOperationException	if the underlying store has no array.

- virtual int **arrayOffset** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

The offset into the backing array where index zero starts.

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
---	--

UnsupportedOperation Exception	if the underlying store has no array.
-----------------------------------	---------------------------------------

- virtual ShortBuffer \* **asReadOnlyBuffer** () const

*Creates a new, read-only short buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent. If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method. The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.*

#### Returns

*The new, read-only short buffer which the caller then owns.*

- virtual ShortBuffer & **compact** () throw ( decaf::nio::ReadOnlyBufferException )

*Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 941) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 940) - 1 is copied to index  $n = \text{limit}()$  (p. 940) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded. The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.*

#### Returns

*a reference to this **ShortBuffer** (p. 3560).*

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only.
---	------------------------------

- virtual ShortBuffer \* **duplicate** ()

*Creates a new short buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent. The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.*

#### Returns

*a new short **Buffer** (p. 936) which the caller owns.*

- virtual short **get** () throw ( decaf::nio::BufferUnderflowException )

*Relative get method.*

*Reads the value at this buffer's current position, and then increments the position.*

**Returns**

*the short at the current position.*

**Exceptions**

<b>BufferUnderflowException</b> (p. 967)	<i>if there no more data to return.</i>
---	---

- virtual short **get** (int index) const throw ( lang::exceptions::IndexOutOfBoundsException )

*Absolute get method.*

*Reads the value at the given index.*

**Parameters**

index	<i>The index in the <b>Buffer</b> (p. 936) where the short is to be read.</i>
-------	---

**Returns**

*the short that is located at the given index.*

**Exceptions**

IndexOutOfBoundsException	<i>if index is not smaller than the buffer's limit, or the index is negative.</i>
---------------------------	---

- virtual bool **hasArray** () const

*Tells whether or not this buffer is backed by an accessible short array.*

*If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.*

**Returns**

*true if, and only if, this buffer is backed by an array and is not read-only.*

- virtual bool **isReadOnly** () const

*Tells whether or not this buffer is read-only.*

**Returns**

*true if, and only if, this buffer is read-only.*

- virtual ShortBuffer & **put** (short value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes the given shorts into this buffer at the current position, and then increments the position.*

**Parameters**

value	<i>The shorts value to be written.</i>
-------	--

**Returns**

*a reference to this buffer.*

*Exceptions*

<b>BufferOverflowException</b> (p. 964)	<i>if this buffer's current position is not smaller than its limit.</i>
<b>ReadOnlyBufferException</b> (p. 3260)	<i>if this buffer is read-only.</i>

- virtual ShortBuffer & **put** (int index, short value) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes the given shorts into this buffer at the given index.*

*Parameters*

index	<i>The position in the <b>Buffer</b> (p. 936) to write the data.</i>
value	<i>The shorts to write.</i>

*Returns*

*a reference to this buffer.*

*Exceptions*

IndexOutOfBoundsException	<i>if index greater than the buffer's limit minus the size of the type being written.</i>
<b>ReadOnlyBufferException</b> (p. 3260)	<i>if this buffer is read-only.</i>

- virtual ShortBuffer \* **slice** () const

*Creates a new **ShortBuffer** (p. 3560) whose content is a shared subsequence of this buffer's content.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

*Returns*

*the newly create **ShortBuffer** (p. 3560) which the caller owns.*

**Protected Member Functions**

- virtual void **setReadOnly** (bool value)

*Sets this **ShortArrayBuffer** (p. 3548) as Read-Only.*

### 6.734.1 Constructor & Destructor Documentation

**6.734.1.1** `decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer ( int size, bool readOnly = false ) throw ( decaf::lang::exceptions::IllegalArgumentException )`

Creates a **ShortArrayBuffer** (p. 3548) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

#### Parameters

<i>size</i>	The size of the array, this is the limit we read and write to.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>IllegalArgumentException</i>	if the capacity value is negative.
---------------------------------	------------------------------------

**6.734.1.2** `decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer ( short * array, int size, int offset, int length, bool readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a **ShortArrayBuffer** (p. 3548) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

#### Parameters

<i>array</i>	The actual array to wrap.
<i>size</i>	The size of the given array.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if offset is greater than array capacity.

```

6.734.1.3  decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (
            const decaf::lang::Pointer< ByteArrayAdapter >
            & array, int offset, int length, bool readOnly = false )
            throw ( decaf::lang::exceptions::NullPointerException,
            decaf::lang::exceptions::IndexOutOfBoundsException )

```

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **ShortArrayBuffer** (p.3548) will be that of the remaining capacity of the passed buffer.

#### Parameters

<i>array</i>	The ByteArrayAdapter to wrap.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>NullPointerException</i>	if array is NULL
<i>IndexOutOfBoundsException</i>	if offset + length is greater than array size.

```

6.734.1.4  decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer ( const ShortArrayBuffer
            & other )

```

Create a **ShortArrayBuffer** (p. 3548) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.

#### Parameters

<i>other</i>	The <b>ShortArrayBuffer</b> (p. 3548) this one is to mirror.
--------------	--

```

6.734.1.5  virtual decaf::internal::nio::ShortArrayBuffer::~ShortArrayBuffer ( )
            [virtual]

```

#### 6.734.2 Member Function Documentation

```

6.734.2.1  virtual short* decaf::internal::nio::ShortArrayBuffer::array ( ) throw
            ( decaf::lang::exceptions::UnsupportedOperationException,
            decaf::nio::ReadOnlyBufferException ) [virtual]

```

Returns the short array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

the array that backs this **Buffer** (p. 936)

### Exceptions

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<i><b>UnsupportedOperationException</b></i>	if the underlying store has no array.

Implements **decaf::nio::ShortBuffer** (p. 3563).

**6.734.2.2** `virtual int decaf::internal::nio::ShortArrayBuffer::arrayOffset ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

The offset into the backing array where index zero starts.

### Exceptions

<i><b>ReadOnlyBufferException</b></i> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<i><b>UnsupportedOperationException</b></i>	if the underlying store has no array.

Implements **decaf::nio::ShortBuffer** (p. 3564).

**6.734.2.3** `virtual ShortBuffer* decaf::internal::nio::ShortArrayBuffer::asReadOnlyBuffer ( ) const [virtual]`

Creates a new, read-only short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and

will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

### Returns

The new, read-only short buffer which the caller then owns.

Implements **decaf::nio::ShortBuffer** (p. 3564).

**6.734.2.4** `virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::compact ( ) throw ( decaf::nio::ReadOnlyBufferException )` [virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 941) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 940) - 1 is copied to index  $n = \text{limit}()$  (p. 940) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

### Returns

a reference to this **ShortBuffer** (p. 3560).

### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.
--	------------------------------

Implements **decaf::nio::ShortBuffer** (p. 3565).

**6.734.2.5** `virtual ShortBuffer* decaf::internal::nio::ShortArrayBuffer::duplicate ( )` [virtual]

Creates a new short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.



The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

a new short **Buffer** (p. 936) which the caller owns.

Implements **decaf::nio::ShortBuffer** (p. 3565).

**6.734.2.6** virtual short decaf::internal::nio::ShortArrayBuffer::get ( ) throw ( decaf::nio::BufferUnderflowException ) [virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

### Returns

the short at the current position.

### Exceptions

<i>BufferUnderflowException</i> (p. 967)	if there no more data to return.
---	----------------------------------

Implements **decaf::nio::ShortBuffer** (p. 3567).

**6.734.2.7** virtual short decaf::internal::nio::ShortArrayBuffer::get ( int index ) const throw ( lang::exceptions::IndexOutOfBoundsException ) [virtual]

Absolute get method.

Reads the value at the given index.

### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the short is to be read.
--------------	--

### Returns

the short that is located at the given index.

### Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or the index is negative.
----------------------------------	--

Implements **decaf::nio::ShortBuffer** (p. 3566).

**6.734.2.8** `virtual bool decaf::internal::nio::ShortArrayBuffer::hasArray ( ) const` `[inline, virtual]`

Tells whether or not this buffer is backed by an accessible short array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

#### Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implements **decaf::nio::ShortBuffer** (p. 3568).

**6.734.2.9** `virtual bool decaf::internal::nio::ShortArrayBuffer::isReadOnly ( ) const` `[inline, virtual]`

Tells whether or not this buffer is read-only.

#### Returns

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 940).

**6.734.2.10** `virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::put ( int index, short value ) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )` `[virtual]`

Writes the given shorts into this buffer at the given index.

#### Parameters

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data.
<i>value</i>	The shorts to write.

#### Returns

a reference to this buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written.
<i>ReadOnlyBufferException</i> (p. 3260)	if this buffer is read-only.

Implements **decaf::nio::ShortBuffer** (p. 3569).

**6.734.2.11** `virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::put ( short value ) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )` [virtual]

Writes the given shorts into this buffer at the current position, and then increments the position.

#### Parameters

<i>value</i>	The shorts value to be written.
--------------	---------------------------------

#### Returns

a reference to this buffer.

#### Exceptions

<b><i>BufferOverflowException</i></b> (p. 964)	if this buffer's current position is not smaller than its limit.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.

Implements **decaf::nio::ShortBuffer** (p. 3568).

**6.734.2.12** `virtual void decaf::internal::nio::ShortArrayBuffer::setReadOnly ( bool value )`  
[inline, protected, virtual]

Sets this **ShortArrayBuffer** (p. 3548) as Read-Only.

#### Parameters

<i>value</i>	Boolean value, true if this buffer is to be read-only, false otherwise.
--------------	---

**6.734.2.13** `virtual ShortBuffer* decaf::internal::nio::ShortArrayBuffer::slice ( ) const`  
[virtual]

Creates a new **ShortBuffer** (p. 3560) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns**

the newly create **ShortBuffer** (p. 3560) which the caller owns.

Implements **decaf::nio::ShortBuffer** (p. 3571).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**ShortArrayBuffer.h**

**6.735 decaf::nio::ShortBuffer Class Reference**

This class defines four categories of operations upon short buffers:

```
#include <src/main/decaf/nio/ShortBuffer.h>
```

Inheritance diagram for decaf::nio::ShortBuffer:

**Public Member Functions**

- virtual **~ShortBuffer** ()
- virtual std::string **toString** () const
- virtual short \* **array** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )  
*Returns the short array that backs this buffer (optional operation).*
- virtual int **arrayOffset** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )  
*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*
- virtual **ShortBuffer** \* **asReadOnlyBuffer** () const =0  
*Creates a new, read-only short buffer that shares this buffer's content.*
- virtual **ShortBuffer** & **compact** ()=0 throw ( ReadOnlyBufferException )  
*Compacts this buffer.*
- virtual **ShortBuffer** \* **duplicate** ()=0  
*Creates a new short buffer that shares this buffer's content.*
- virtual short **get** ()=0 throw ( BufferUnderflowException )  
*Relative get method.*
- virtual short **get** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Absolute get method.*

- **ShortBuffer** & **get** (std::vector< short > buffer) throw ( BufferUnderflowException )

*Relative bulk get method.*

- **ShortBuffer** & **get** (short \*buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*Relative bulk get method.*

- virtual bool **hasArray** () const =0

*Tells whether or not this buffer is backed by an accessible short array.*

- **ShortBuffer** & **put** (**ShortBuffer** &src) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )

*This method transfers the shorts remaining in the given source buffer into this buffer.*

- **ShortBuffer** & **put** (const short \*buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*This method transfers shorts into this buffer from the given source array.*

- **ShortBuffer** & **put** (std::vector< short > &buffer) throw ( BufferOverflowException, ReadOnlyBufferException )

*This method transfers the entire content of the given source shorts array into this buffer.*

- virtual **ShortBuffer** & **put** (short value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )

*Writes the given shorts into this buffer at the current position, and then increments the position.*

- virtual **ShortBuffer** & **put** (int index, short value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )

*Writes the given shorts into this buffer at the given index.*

- virtual **ShortBuffer** \* **slice** () const =0

*Creates a new **ShortBuffer** (p. 3560) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **ShortBuffer** &value) const

- virtual bool **equals** (const **ShortBuffer** &value) const

- virtual bool **operator==** (const **ShortBuffer** &value) const

- virtual bool **operator**< (const **ShortBuffer** &value) const

### Static Public Member Functions

- static **ShortBuffer** \* **allocate** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Allocates a new Double buffer.*

- static **ShortBuffer** \* **wrap** (short \*array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new **ShortBuffer** (p. 3560).*

- static **ShortBuffer** \* **wrap** (std::vector< short > &buffer)

*Wraps the passed STL short Vector in a **ShortBuffer** (p. 3560).*

### Protected Member Functions

- **ShortBuffer** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Creates a **ShortBuffer** (p. 3560) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

#### 6.735.1 Detailed Description

This class defines four categories of operations upon short buffers: o Absolute and relative get and put methods that read and write single shorts; o Relative bulk get methods that transfer contiguous sequences of shorts from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of shorts from a short array or some other short buffer into this buffer o Methods for compacting, duplicating, and slicing a short buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing short array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

## 6.735.2 Constructor & Destructor Documentation

**6.735.2.1** `decaf::nio::ShortBuffer::ShortBuffer ( int capacity ) throw ( decaf::lang::exceptions::IllegalArgumentException ) [protected]`

Creates a **ShortBuffer** (p. 3560) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

### Parameters

<i>capacity</i>	The size and limit of the <b>Buffer</b> (p. 936) in doubles
-----------------	---

### Exceptions

<i>IllegalArgumentEx-ception</i>	if capacity is negative.
----------------------------------	--------------------------

**6.735.2.2** `virtual decaf::nio::ShortBuffer::~~ShortBuffer ( ) [inline, virtual]`

## 6.735.3 Member Function Documentation

**6.735.3.1** `static ShortBuffer* decaf::nio::ShortBuffer::allocate ( int capacity ) throw ( decaf::lang::exceptions::IllegalArgumentException ) [static]`

Allocates a new Double buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

### Parameters

<i>capacity</i>	The size of the Double buffer in shorts.
-----------------	--

### Returns

the **ShortBuffer** (p. 3560) that was allocated, caller owns.

**6.735.3.2** `virtual short* decaf::nio::ShortBuffer::array ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the short array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns**

the array that backs this **Buffer** (p. 936)

**Exceptions**

<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<b><i>UnsupportedOperationException</i></b>	if the underlying store has no array.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3554).

**6.735.3.3** `virtual int decaf::nio::ShortBuffer::arrayOffset ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns**

The offset into the backing array where index zero starts.

**Exceptions**

<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this <b>Buffer</b> (p. 936) is read only.
<b><i>UnsupportedOperationException</i></b>	if the underlying store has no array.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3555).

**6.735.3.4** `virtual ShortBuffer* decaf::nio::ShortBuffer::asReadOnlyBuffer ( ) const [pure virtual]`

Creates a new, read-only short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.



The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

### Returns

The new, read-only short buffer which the caller then owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3555).

**6.735.3.5** `virtual ShortBuffer& decaf::nio::ShortBuffer::compact ( ) throw ( ReadOnlyBufferException )` [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 941) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 940) - 1 is copied to index  $n = \text{limit}()$  (p. 940) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

### Returns

a reference to this **ShortBuffer** (p. 3560).

### Exceptions

<b>ReadOnlyBufferException</b> (p. 3260)	if this buffer is read-only.
---	------------------------------

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3556).

**6.735.3.6** `virtual int decaf::nio::ShortBuffer::compareTo ( const ShortBuffer & value ) const` [virtual]

**6.735.3.7** `virtual ShortBuffer* decaf::nio::ShortBuffer::duplicate ( )` [pure virtual]

Creates a new short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns**

a new short **Buffer** (p. 936) which the caller owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3556).

**6.735.3.8** `virtual bool decaf::nio::ShortBuffer::equals ( const ShortBuffer & value ) const`  
[virtual]

**6.735.3.9** `ShortBuffer& decaf::nio::ShortBuffer::get ( std::vector< short > buffer ) throw ( BufferUnderflowException )`

Relative bulk get method.

This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize( N )` before calling this get method.

**Returns**

a reference to this **Buffer** (p. 936).

**Exceptions**

<i><b>BufferUnderflowException</b></i> (p. 967)	if there are fewer than length shorts remaining in this buffer.
--	---

**6.735.3.10** `virtual short decaf::nio::ShortBuffer::get ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [pure virtual]

Absolute get method.

Reads the value at the given index.

**Parameters**

<i>index</i>	The index in the <b>Buffer</b> (p. 936) where the short is to be read.
--------------	--

**Returns**

the short that is located at the given index.

**Exceptions**

<i><b>IndexOutOfBoundsException</b></i>	if index is not smaller than the buffer's limit, or the index is negative.
---	--

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3557).

**6.735.3.11 ShortBuffer& decaf::nio::ShortBuffer::get ( short \* *buffer*, int *size*, int *offset*, int *length* ) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )**

Relative bulk get method.

This method transfers shorts from this buffer into the given destination array. If there are fewer shorts remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 941), then no bytes are transferred and a **BufferUnderflowException** (p. 967) is thrown.

Otherwise, this method copies `length` shorts from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

#### Parameters

<i>buffer</i>	The pointer to an allocated buffer to fill.
<i>size</i>	The size of the buffer provided.
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The amount of data to put in the passed buffer.

#### Returns

a reference to this **Buffer** (p. 936).

#### Exceptions

<b>BufferUnderflowException</b> (p. 967)	if there are fewer than <code>length</code> shorts remaining in this buffer
<b>NullPointerException</b>	if the passed buffer is null.
<b>IndexOutOfBoundsException</b>	if the preconditions of <code>size</code> , <code>offset</code> , or <code>length</code> are not met.

**6.735.3.12 virtual short decaf::nio::ShortBuffer::get ( ) throw ( BufferUnderflowException ) [pure virtual]**

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

#### Returns

the short at the current position.

**Exceptions**

<b><i>BufferUnderflowException</i></b> (p. 967)	if there no more data to return.
--	----------------------------------

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3557).

**6.735.3.13** `virtual bool decaf::nio::ShortBuffer::hasArray ( ) const` [pure virtual]

Tells whether or not this buffer is backed by an accessible short array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns**

true if, and only if, this buffer is backed by an array and is not read-only.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3558).

**6.735.3.14** `virtual bool decaf::nio::ShortBuffer::operator< ( const ShortBuffer & value ) const` [virtual]

**6.735.3.15** `virtual bool decaf::nio::ShortBuffer::operator== ( const ShortBuffer & value ) const` [virtual]

**6.735.3.16** `virtual ShortBuffer& decaf::nio::ShortBuffer::put ( short value ) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes the given shorts into this buffer at the current position, and then increments the position.

**Parameters**

<i>value</i>	The shorts value to be written.
--------------	---------------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 964)	if this buffer's current position is not smaller than its limit.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3559).

**6.735.3.17** virtual **ShortBuffer&** **decaf::nio::ShortBuffer::put** ( int *index*, short *value* ) throw ( **decaf::lang::exceptions::IndexOutOfBoundsException**, **ReadOnlyBufferException** ) [pure virtual]

Writes the given shorts into this buffer at the given index.

#### Parameters

<i>index</i>	The position in the <b>Buffer</b> (p. 936) to write the data.
<i>value</i>	The shorts to write.

#### Returns

a reference to this buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written.
<i>ReadOnlyBufferException</i> (p. 3260)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3558).

**6.735.3.18** **ShortBuffer&** **decaf::nio::ShortBuffer::put** ( const short \* *buffer*, int *size*, int *offset*, int *length* ) throw ( **BufferOverflowException**, **ReadOnlyBufferException**, **decaf::lang::exceptions::IndexOutOfBoundsException**, **decaf::lang::exceptions::NullPointerException** )

This method transfers shorts into this buffer from the given source array.

If there are more shorts to be copied from the array than remain in this buffer, that is, if *length* > **remaining()** (p. 941), then no shorts are transferred and a **BufferOverflowException** (p. 964) is thrown.

Otherwise, this method copies *length* bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by *length*.

#### Parameters

<i>buffer</i>	The array from which shorts are to be read.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The offset within the array of the first char to be read.
<i>length</i>	The number of shorts to be read from the given array.

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 964)	if there is insufficient space in this buffer
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only
<b><i>NullPointerException</i></b>	if the passed buffer is null.
<b><i>IndexOutOfBoundsException</i></b>	if the preconditions of size, offset, or length are not met.

#### 6.735.3.19 **ShortBuffer& decaf::nio::ShortBuffer::put ( std::vector< short > & buffer ) throw ( BufferOverflowException, ReadOnlyBufferException )**

This method transfers the entire content of the given source shorts array into this buffer.

This is the same as calling put( &buffer[0], 0, buffer.size()).

**Parameters**

<i>buffer</i>	The buffer whose contents are copied to this <b>ShortBuffer</b> (p. 3560).
---------------	--

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 964)	if there is insufficient space in this buffer.
<b><i>ReadOnlyBufferException</i></b> (p. 3260)	if this buffer is read-only.

#### 6.735.3.20 **ShortBuffer& decaf::nio::ShortBuffer::put ( ShortBuffer & src ) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )**

This method transfers the shorts remaining in the given source buffer into this buffer.

If there are more shorts remaining in the source buffer than in this buffer, that is, if src.remaining() > **remaining()** (p. 941), then no shorts are transferred and a **BufferOverflowException** (p. 964) is thrown.

Otherwise, this method copies  $n = \text{src.remaining}()$  shorts from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by  $n$ .

### Parameters

<i>src</i>	The buffer to take shorts from an place in this one.
------------	--

### Returns

a reference to this buffer.

### Exceptions

<b><i>BufferOverflowException</i></b> (p. 964)	if there is insufficient space in this buffer for the remaining shorts in the source buffer.
<b><i>IllegalArgumentEx-ception</i></b>	if the source buffer is this buffer.
<b><i>ReadOnlyBufferEx-ception</i></b> (p. 3260)	if this buffer is read-only.

**6.735.3.21** `virtual ShortBuffer* decaf::nio::ShortBuffer::slice ( ) const` [pure virtual]

Creates a new **ShortBuffer** (p. 3560) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the newly create **ShortBuffer** (p. 3560) which the caller owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3559).

**6.735.3.22** `virtual std::string decaf::nio::ShortBuffer::toString ( ) const` [virtual]

### Returns

a std::string describing this object

**6.735.3.23** `static ShortBuffer* decaf::nio::ShortBuffer::wrap ( short * array, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [static]`

Wraps the passed buffer with a new **ShortBuffer** (p. 3560).

The new buffer will be backed by the given short array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>array</i>	The array that will back the new buffer.
<i>size</i>	The size of the passed in array.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

#### Returns

a new **ShortBuffer** (p. 3560) that is backed by buffer, caller owns.

#### Exceptions

<i>NullPointerException</i>	if the array pointer is NULL.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

**6.735.3.24** `static ShortBuffer* decaf::nio::ShortBuffer::wrap ( std::vector< short > & buffer ) [static]`

Wraps the passed STL short Vector in a **ShortBuffer** (p. 3560).

The new buffer will be backed by the given short array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling <code>vector.resize( N )</code> .
---------------	--

#### Returns

a new **ShortBuffer** (p. 3560) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/ShortBuffer.h`



## 6.736 activemq::commands::ShutdownInfo Class Reference

```
#include <src/main/activemq/commands/ShutdownInfo.h>
```

Inheritance diagram for activemq::commands::ShutdownInfo:

### Public Member Functions

- **ShutdownInfo** ()
- virtual **~ShutdownInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ShutdownInfo** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual bool **isShutdownInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_SHUTDOWNINFO** = 11

## 6.736.1 Constructor & Destructor Documentation

6.736.1.1 `activemq::commands::ShutdownInfo::ShutdownInfo ( )`

6.736.1.2 `virtual activemq::commands::ShutdownInfo::~~ShutdownInfo ( ) [virtual]`

## 6.736.2 Member Function Documentation

6.736.2.1 `virtual ShutdownInfo* activemq::commands::ShutdownInfo::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1714).

6.736.2.2 `virtual void activemq::commands::ShutdownInfo::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

6.736.2.3 `virtual bool activemq::commands::ShutdownInfo::equals ( const DataStructure * value ) const [virtual]`

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

**6.736.2.4** `virtual unsigned char activemq::commands::ShutdownInfo::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataSet** (p. 1713) type copy.

Implements **activemq::commands::DataSet** (p. 1717).

**6.736.2.5** `virtual bool activemq::commands::ShutdownInfo::isShutdownInfo ( ) const [inline, virtual]`

#### Returns

an answer of true to the **isShutdownInfo()** (p. 3575) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

**6.736.2.6** `virtual std::string activemq::commands::ShutdownInfo::toString ( ) const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

**6.736.2.7** `virtual Pointer<Command> activemq::commands::ShutdownInfo::visit ( activemq::state::CommandVisitor * visitor ) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

### 6.736.3 Field Documentation

6.736.3.1 `const unsigned char activemq::commands::ShutdownInfo::ID_SHUTDOWNINFO = 11` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ShutdownInfo.h`

## 6.737 `activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3576).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ShutdownInfoMar
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller`:

### Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual `~ShutdownInfoMarshaller` ()
- virtual `commands::DataStructure * createObject ()` const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataInputStream *dataIn`, `utils::BooleanStream *bs`) throw ( `decaf::io::IOException` )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `utils::BooleanStream *bs`) throw ( `decaf::io::IOException` )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataOutputStream *dataOut`, `utils::BooleanStream *bs`) throw ( `decaf::io::IOException` )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.737.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3576). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.737.2 Constructor & Destructor Documentation

**6.737.2.1** **activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::ShutdownInfoMarshaller**  
( ) [inline]

**6.737.2.2** **virtual activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller**  
( ) [inline, virtual]

### 6.737.3 Member Function Documentation

**6.737.3.1** **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.737.3.2** **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

```
6.737.3.3  virtual void activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 801).

```
6.737.3.4  virtual void activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 802).

6.737.3.5 virtual int activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::tightMarshal1  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller**  
(p. 803).

6.737.3.6 virtual void activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::tightMarshal2  
( OpenWireFormat \* *wireFormat*, commands::DataStructure  
\* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*,  
utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller**  
(p. 804).

```

6.737.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 806).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ShutdownInfoMarshaller.h**

## 6.738 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3580).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMar
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller**:

#### Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*



- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.738.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3580). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.738.2 Constructor & Destructor Documentation

**6.738.2.1** **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::ShutdownInfoMarshaller**  
( ) [inline]

**6.738.2.2** **virtual activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller**  
( ) [inline, virtual]

### 6.738.3 Member Function Documentation

**6.738.3.1** **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.738.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.738.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 808).

**6.738.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 809).

**6.738.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 810).

**6.738.3.6** `virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 811).

**6.738.3.7** `virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 813).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMarshaller.h`

## 6.739 **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3584).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMar
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller**:

## Public Member Functions

- **ShutdownInfoMarshaller ()**
- virtual **~ShutdownInfoMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.739.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3584). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.739.2 Constructor & Destructor Documentation

6.739.2.1 `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::ShutdownInfoMarshaller ( )` `[inline]`

6.739.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller ( )` `[inline, virtual]`

## 6.739.3 Member Function Documentation

6.739.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::createObject ( )`  
`const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.739.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::getDataStructureType ( )` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.739.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut )` `throw ( decaf::io::IOException )` `[virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 787).

**6.739.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 788).

**6.739.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 789).

```
6.739.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 790).

```
6.739.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 792).

The documentation for this class was generated from the following file:



- src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMarshaller.h

## 6.740 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3589).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ShutdownInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller:

### Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.740.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3589). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.740.2 Constructor & Destructor Documentation

6.740.2.1 **activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::ShutdownInfoMarshaller**  
( ) [inline]

6.740.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller**  
( ) [inline, virtual]

### 6.740.3 Member Function Documentation

6.740.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.740.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.740.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) **throw** ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 794).

**6.740.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 795).

**6.740.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 796).

```
6.740.3.6  virtual void activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 797).

```
6.740.3.7  virtual void activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ShutdownInfoMarshaller.h**

## 6.741 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3593).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller**:

## Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.741.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3593). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.741.2 Constructor & Destructor Documentation

- 6.741.2.1 **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::ShutdownInfoMarshaller**  
 ( ) [inline]
- 6.741.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller**  
 ( ) [inline, virtual]

### 6.741.3 Member Function Documentation

- 6.741.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

- 6.741.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::getDataStructureType**  
 ( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

## 6.741 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller

### Class Reference 3599

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.741.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 772).

**6.741.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 774).

```
6.741.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 775).

```
6.741.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 776).



## 6.742 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller

### Class Reference

3601

6.741.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ShutdownInfoMarshaller.h**

## 6.742 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3597).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller:

#### Public Member Functions

- ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.742.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3597). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.742.2 Constructor & Destructor Documentation

6.742.2.1 **activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::ShutdownInfoMarshaller** ( ) [inline]

6.742.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller** ( ) [inline, virtual]

### 6.742.3 Member Function Documentation

6.742.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.742.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::getDataStructureType ( ) const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.742.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 780).

**6.742.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 781).

```
6.742.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 782).

```
6.742.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 783).

**6.742.3.7** virtual void **activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h`

**6.743 decaf::security::SignatureException Class Reference**

```
#include <src/main/decaf/security/SignatureException.h>
```

Inheritance diagram for **decaf::security::SignatureException**:

**Public Member Functions**

- **SignatureException** () throw ()

*Default Constructor.*

- **SignatureException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **SignatureException** (const **SignatureException** &ex) throw ()  
*Copy Constructor.*
- **SignatureException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **SignatureException** (const std::exception \*cause) throw ()  
*Constructor.*
- **SignatureException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **SignatureException \* clone** () const  
*Clones this exception.*
- virtual **~SignatureException** () throw ()

## 6.743.1 Constructor & Destructor Documentation

6.743.1.1 **decaf::security::SignatureException::SignatureException ( ) throw ()** [inline]

Default Constructor.

6.743.1.2 **decaf::security::SignatureException::SignatureException ( const Exception & ex ) throw ()** [inline]

Conversion Constructor from some other Exception.

### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.743.1.3 **decaf::security::SignatureException::SignatureException ( const SignatureException & ex ) throw ()** [inline]

Copy Constructor.

**Parameters**

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.743.1.4** decaf::security::SignatureException::SignatureException ( const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.  
Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.743.1.5** decaf::security::SignatureException::SignatureException ( const std::exception \* *cause* ) throw () [inline]

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.743.1.6** decaf::security::SignatureException::SignatureException ( const char \* *file*, const int *lineNumber*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.  
Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
...	list of primitives that are formatted into the message

**6.743.1.7** `virtual decaf::security::SignatureException::~~SignatureException ( ) throw ()`  
`[inline, virtual]`

## 6.743.2 Member Function Documentation

**6.743.2.1** `virtual SignatureException* decaf::security::SignatureException::clone ( ) const`  
`[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 2037).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/SignatureException.h`

## 6.744 decaf::util::logging::SimpleFormatter Class Reference

Print a brief summary of the **LogRecord** (p. 2487) in a human readable format.

```
#include <src/main/decaf/util/logging/SimpleFormatter.h>
```

Inheritance diagram for `decaf::util::logging::SimpleFormatter`:

### Public Member Functions

- **SimpleFormatter** ()
- virtual **~SimpleFormatter** ()
- virtual `std::string format (const LogRecord &record) const`

*Format the given log record and return the formatted string.*

### 6.744.1 Detailed Description

Print a brief summary of the **LogRecord** (p. 2487) in a human readable format. The summary will typically be 1 or 2 lines.

### Since

1.0



## 6.744.2 Constructor & Destructor Documentation

6.744.2.1 decaf::util::logging::SimpleFormatter::SimpleFormatter ( )

6.744.2.2 virtual decaf::util::logging::SimpleFormatter::~SimpleFormatter ( ) [virtual]

## 6.744.3 Member Function Documentation

6.744.3.1 virtual std::string decaf::util::logging::SimpleFormatter::format ( const **LogRecord** & *record* ) const [virtual]

Format the given log record and return the formatted string.

### Parameters

<i>record</i>	The Log Record to Format.
---------------	---------------------------

Implements **decaf::util::logging::Formatter** (p. 2028).

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**SimpleFormatter.h**

## 6.745 decaf::util::logging::SimpleLogger Class Reference

```
#include <src/main/decaf/util/logging/SimpleLogger.h>
```

### Public Member Functions

- **SimpleLogger** (const std::string &name)  
*Constructor.*
- virtual ~**SimpleLogger** ()  
*Destructor.*
- virtual void **mark** (const std::string &message)  
*Log a Mark Block **Level** (p. 2403) Log.*
- virtual void **debug** (const std::string &file, const int line, const std::string &message)  
*Log a Debug **Level** (p. 2403) Log.*
- virtual void **info** (const std::string &file, const int line, const std::string &message)  
*Log a Informational **Level** (p. 2403) Log.*

- virtual void **warn** (const std::string &file, const int line, const std::string &message)

*Log a Warning **Level** (p. 2403) Log.*

- virtual void **error** (const std::string &file, const int line, const std::string &message)

*Log a Error **Level** (p. 2403) Log.*

- virtual void **fatal** (const std::string &file, const int line, const std::string &message)

*Log a Fatal **Level** (p. 2403) Log.*

- virtual void **log** (const std::string &message)

*No-frills log.*

## 6.745.1 Constructor & Destructor Documentation

### 6.745.1.1 decaf::util::logging::SimpleLogger::SimpleLogger ( const std::string & name )

Constructor.

### 6.745.1.2 virtual decaf::util::logging::SimpleLogger::~SimpleLogger ( ) [virtual]

Destructor.

## 6.745.2 Member Function Documentation

### 6.745.2.1 virtual void decaf::util::logging::SimpleLogger::debug ( const std::string & file, const int line, const std::string & message ) [virtual]

Log a Debug **Level** (p. 2403) Log.

### 6.745.2.2 virtual void decaf::util::logging::SimpleLogger::error ( const std::string & file, const int line, const std::string & message ) [virtual]

Log a Error **Level** (p. 2403) Log.

### 6.745.2.3 virtual void decaf::util::logging::SimpleLogger::fatal ( const std::string & file, const int line, const std::string & message ) [virtual]

Log a Fatal **Level** (p. 2403) Log.

**6.745.2.4** virtual void decaf::util::logging::SimpleLogger::info ( const std::string & *file*, const int *line*, const std::string & *message* ) [virtual]

Log a Informational **Level** (p. 2403) Log.

**6.745.2.5** virtual void decaf::util::logging::SimpleLogger::log ( const std::string & *message* ) [virtual]

No-frills log.

**6.745.2.6** virtual void decaf::util::logging::SimpleLogger::mark ( const std::string & *message* ) [virtual]

Log a Mark Block **Level** (p. 2403) Log.

**6.745.2.7** virtual void decaf::util::logging::SimpleLogger::warn ( const std::string & *file*, const int *line*, const std::string & *message* ) [virtual]

Log a Warning **Level** (p. 2403) Log.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/SimpleLogger.h

## 6.746 decaf::net::Socket Class Reference

```
#include <src/main/decaf/net/Socket.h>
```

Inheritance diagram for decaf::net::Socket:

### Public Member Functions

- **Socket** ()  
*Creates an unconnected **Socket** (p. 3607) using the set **SocketImplFactory** (p. 3644) or if non is set than the default **SocketImpl** type is created.*
- **Socket** (**SocketImpl** \*impl)  
*Creates a **Socket** (p. 3607) wrapping the provided **SocketImpl** (p. 3635) instance, this **Socket** (p. 3607) is considered unconnected.*
- **Socket** (const **InetAddress** \*address, int port)  
*Creates a new **Socket** (p. 3607) instance and connects it to the given address and port.*

- **Socket** (const **InetAddress** \*address, int port, const **InetAddress** \*localAddress, int localPort)  
*Creates a new **Socket** (p. 3607) instance and connects it to the given address and port.*
- **Socket** (const std::string &host, int port)  
*Creates a new **Socket** (p. 3607) instance and connects it to the given host and port.*
- **Socket** (const std::string &host, int port, const **InetAddress** \*localAddress, int localPort)  
*Creates a new **Socket** (p. 3607) instance and connects it to the given host and port.*
- virtual ~**Socket** ()
- virtual void **bind** (const std::string &ipaddress, int port) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Binds this **Socket** (p. 3607) to the given local address and port.*
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes the **Socket** (p. 3607).*
- virtual void **connect** (const std::string &host, int port) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Connects to the specified destination.*
- virtual void **connect** (const std::string &host, int port, int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Connects to the specified destination, with a specified timeout value.*
- bool **isConnected** () const  
*Indicates whether or not this socket is connected to an end point.*
- bool **isClosed** () const
- bool **isBound** () const
- bool **isInputShutdown** () const
- bool **isOutputShutdown** () const
- virtual **decaf::io::InputStream** \* **getInputStream** () throw ( decaf::io::IOException )  
*Gets the **InputStream** for this socket if its connected.*
- virtual **decaf::io::OutputStream** \* **getOutputStream** () throw ( decaf::io::IOException )  
*Gets the **OutputStream** for this socket if it is connected.*
- int **getPort** () const  
*Gets the on the remote host this **Socket** (p. 3607) is connected to.*
- int **getLocalPort** () const  
*Gets the local port the socket is bound to.*

- std::string **getInetAddress** () const  
*Returns the address to which the socket is connected.*
- std::string **getLocalAddress** () const  
*Gets the local address to which the socket is bound.*
- virtual void **shutdownInput** () throw ( decaf::io::IOException )  
*Shuts down the InputStream for this socket essentially marking it as EOF.*
- virtual void **shutdownOutput** () throw ( decaf::io::IOException )  
*Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to OutputStream::write will throw an IOException.*
- virtual int **getSoLinger** () const throw ( SocketException )  
*Gets the linger time for the socket, SO\_LINGER.*
- virtual void **setSoLinger** (bool state, int timeout) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )  
*Sets the linger time (SO\_LINGER) using a specified time value, this limits of this value are platform specific.*
- virtual bool **getKeepAlive** () const throw ( SocketException )  
*Gets the keep alive flag for this socket, SO\_KEEPAIVE.*
- virtual void **setKeepAlive** (bool keepAlive) throw ( SocketException )  
*Enables/disables the keep alive flag for this socket, SO\_KEEPAIVE.*
- virtual int **getReceiveBufferSize** () const throw ( SocketException )  
*Gets the receive buffer size for this socket, SO\_RCVBUF.*
- virtual void **setReceiveBufferSize** (int size) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )  
*Sets the receive buffer size for this socket, SO\_RCVBUF.*
- virtual bool **getReuseAddress** () const throw ( SocketException )  
*Gets the reuse address flag, SO\_REUSEADDR.*
- virtual void **setReuseAddress** (bool reuse) throw ( SocketException )  
*Sets the reuse address flag, SO\_REUSEADDR.*
- virtual int **getSendBufferSize** () const throw ( SocketException )  
*Gets the send buffer size for this socket, SO\_SNDBUF, this value is used by the platform socket to buffer data written to the socket.*
- virtual void **setSendBufferSize** (int size) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )

*Gets the send buffer size for this socket, `SO_SNDBUF`, this value is used by the platform socket to buffer data written to the socket.*

- virtual int **getSoTimeout** () const throw ( SocketException )  
*Gets the timeout for socket operations, `SO_TIMEOUT`.*
- virtual void **setSoTimeout** (int timeout) throw ( SocketException, decaf::lang::exceptions::IllegalArgument )  
*Sets the timeout for socket operations, `SO_TIMEOUT`.*
- virtual bool **getTcpNoDelay** () const throw ( SocketException )  
*Gets the Status of the `TCP_NODELAY` setting for this socket.*
- virtual void **setTcpNoDelay** (bool value) throw ( SocketException )  
*Sets the Status of the `TCP_NODELAY` param for this socket., this setting is used to disable or enable Nagle's algorithm on the **Socket** (p. 3607).*
- virtual int **getTrafficClass** () const throw ( SocketException )  
*Gets the Traffic Class setting for this **Socket** (p. 3607), sometimes referred to as Type of Service setting.*
- virtual void **setTrafficClass** (int value) throw ( SocketException, decaf::lang::exceptions::IllegalArgument )  
*Gets the Traffic Class setting for this **Socket** (p. 3607), sometimes referred to as Type of Service setting.*
- virtual bool **getOOBInline** () const throw ( SocketException )  
*Gets the value of the `OOBINLINE` for this socket.*
- virtual void **setOOBInline** (bool value) throw ( SocketException )  
*Sets the value of the `OOBINLINE` for this socket, by default this option is disabled.*
- virtual void **sendUrgentData** (int data) throw ( decaf::io::IOException )  
*Sends on byte of urgent data to the **Socket** (p. 3607).*
- virtual std::string **toString** () const

### Static Public Member Functions

- static void **setSocketImplFactory** (SocketImplFactory \*factory) throw ( decaf::io::IOException, decaf::net::SocketException )  
*Sets the instance of a **SocketImplFactory** (p. 3644) that the **Socket** (p. 3607) class should use when new instances of this class are created.*

## Protected Member Functions

- void **accepted** ()
- void **initSocketImpl** (const std::string &address, int port, const **InetAddress** \*localAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException )
- void **checkClosed** () const throw ( decaf::io::IOException )
- void **ensureCreated** () const throw ( decaf::io::IOException )

## Protected Attributes

- **SocketImpl** \* **impl**

## Friends

- class **ServerSocket**

### 6.746.1 Detailed Description

Since

1.0

### 6.746.2 Constructor & Destructor Documentation

#### 6.746.2.1 decaf::net::Socket::Socket ( )

Creates an unconnected **Socket** (p. 3607) using the set **SocketImplFactory** (p. 3644) or if non is set than the default **SocketImpl** type is created.

#### 6.746.2.2 decaf::net::Socket::Socket ( **SocketImpl** \* *impl* )

Creates a **Socket** (p. 3607) wrapping the provided **SocketImpl** (p. 3635) instance, this **Socket** (p. 3607) is considered unconnected.

The **Socket** (p. 3607) class takes ownership of this **SocketImpl** (p. 3635) pointer and will delete it when the **Socket** (p. 3607) class is destroyed.

## Parameters

<i>impl</i>	The <b>SocketImpl</b> (p. 3635) instance to wrap.
-------------	---

## Exceptions

<i>NullPointerException</i>	if the passed <b>SocketImpl</b> (p. 3635) is Null.
-----------------------------	--

**6.746.2.3 decaf::net::Socket::Socket ( const InetAddress \* *address*, int *port* )**

Creates a new **Socket** (p. 3607) instance and connects it to the given address and port.

If there is a **SocketImplFactory** (p. 3644) set then the SocketImpl is created using the factory otherwise the default **Socket** (p. 3607) implementation is used.

If the host parameter is empty then the loop back address is used.

**Parameters**

<i>address</i>	The address to connect to.
<i>port</i>	The port number to connect to [0...65535]

**Exceptions**

<b>UnknownHostException</b> (p. 4019)	if the host cannot be resolved.
<b>IOException</b>	if an I/O error occurs while connecting the <b>Socket</b> (p. 3607).
<b>NullPointerException</b>	if the <b>InetAddress</b> (p. 2077) instance is NULL.
<b>IllegalArgumentException</b>	if the port is not in range [0...65535]

**6.746.2.4 decaf::net::Socket::Socket ( const InetAddress \* *address*, int *port*, const InetAddress \* *localAddress*, int *localPort* )**

Creates a new **Socket** (p. 3607) instance and connects it to the given address and port.

If there is a **SocketImplFactory** (p. 3644) set then the SocketImpl is created using the factory otherwise the default **Socket** (p. 3607) implementation is used. The **Socket** (p. 3607) will also bind to the local address and port specified.

**Parameters**

<i>address</i>	The address to connect to.
<i>port</i>	The port number to connect to [0...65535]
<i>localAddress</i>	The IP address on the local machine to bind to.
<i>localPort</i>	The port on the local machine to bind to.

**Exceptions**

<b>UnknownHostException</b> (p. 4019)	if the host cannot be resolved.
<b>IOException</b>	if an I/O error occurs while connecting the <b>Socket</b> (p. 3607).
<b>NullPointerException</b>	if the <b>InetAddress</b> (p. 2077) instance is NULL.



<i>IllegalArgumentEx- ception</i>	if the port if not in range [0...65535]
---------------------------------------	---

#### 6.746.2.5 decaf::net::Socket::Socket ( const std::string & *host*, int *port* )

Creates a new **Socket** (p. 3607) instance and connects it to the given host and port.

If there is a **SocketImplFactory** (p. 3644) set then the SokcetImpl is created using the factory otherwise the default **Socket** (p. 3607) implementation is used.

If the host parameter is empty then the loop back address is used.

##### Parameters

<i>host</i>	The host name or IP address to connect to, empty string means loopback.
<i>port</i>	The port number to connect to [0...65535]

##### Exceptions

<i>UnknownHostException</i> (p. 4019)	if the host cannot be resolved.
<i>IOException</i>	if an I/O error occurs while connecting the <b>Socket</b> (p. 3607).
<i>IllegalArgumentEx- ception</i>	if the port if not in range [0...65535]

#### 6.746.2.6 decaf::net::Socket::Socket ( const std::string & *host*, int *port*, const InetAddress \* *localAddress*, int *localPort* )

Creates a new **Socket** (p. 3607) instance and connects it to the given host and port.

If there is a **SocketImplFactory** (p. 3644) set then the SokcetImpl is created using the factory otherwise the default **Socket** (p. 3607) implementation is used.

If the host parameter is empty then the loop back address is used.

##### Parameters

<i>host</i>	The host name or IP address to connect to, empty string means loopback.
<i>port</i>	The port number to connect to [0...65535]
<i>localAd- dress</i>	The IP address on the local machine to bind to.
<i>localPort</i>	The port on the local machine to bind to.

##### Exceptions

<i>UnknownHostException</i> (p. 4019)	if the host cannot be resolved.
--	---------------------------------

<i>IOException</i>	if an I/O error occurs while connecting the <b>Socket</b> (p. 3607).
<i>IllegalArgumentEx- ception</i>	if the port if not in range [0...65535]

**6.746.2.7** virtual **decaf::net::Socket::~~Socket** ( ) [virtual]

### 6.746.3 Member Function Documentation

**6.746.3.1** void **decaf::net::Socket::accepted** ( ) [protected]

**6.746.3.2** virtual void **decaf::net::Socket::bind** ( const std::string & *ipaddress*, int *port* ) throw ( **decaf::io::IOException**, **decaf::lang::exceptions::IllegalArgumentException** ) [virtual]

Binds this **Socket** (p. 3607) to the given local address and port.

If the **SocketAddress** (p. 3626) value is NULL then the **Socket** (p. 3607) will be bound to an available local address and port.

#### Parameters

<i>ipaddress</i>	The local address and port to bind the socket to.
<i>port</i>	The port on the local machine to bind to.

#### Exceptions

<i>IOException</i>	if an error occurs during the bind operation.
<i>IllegalArgumentEx- ception</i>	if the <b>Socket</b> (p. 3607) can't process the subclass of <b>SocketAddress</b> (p. 3626) that has been provided.

**6.746.3.3** void **decaf::net::Socket::checkClosed** ( ) const throw ( **decaf::io::IOException** ) [protected]

**6.746.3.4** virtual void **decaf::net::Socket::close** ( ) throw ( **decaf::io::IOException** ) [virtual]

Closes the **Socket** (p. 3607).

Once closed a **Socket** (p. 3607) cannot be connected or otherwise operated upon, a new **Socket** (p. 3607) instance must be created.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while closing the <b>Socket</b> (p. 3607).
--------------------	---

Implements **decaf::io::Closeable** (p. 1181).

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2947).

**6.746.3.5** virtual void decaf::net::Socket::connect ( const std::string & *host*, int *port*, int *timeout* ) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException ) [virtual]

Connects to the specified destination, with a specified timeout value.

If a connection to the remote host is not established within the specified timeout interval than an **SocketTimeoutException** (p. 3650) is thrown. A timeout value of zero is treated as an infinite timeout.

#### Parameters

<i>host</i>	The host name or IP address of the remote host to connect to.
<i>port</i>	The port on the remote host to connect to.
<i>timeout</i>	The number of Milliseconds to wait before treating the connection as failed.

#### Exceptions

<i>IOException</i>	Thrown if a failure occurred in the connect.
<i>SocketTimeoutException</i> (p. 3650)	if the timeout for connection is exceeded.
<i>IllegalArgumentEx-ception</i>	if the timeout value is negative or the endpoint is invalid.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2947).

**6.746.3.6** virtual void decaf::net::Socket::connect ( const std::string & *host*, int *port* ) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException ) [virtual]

Connects to the specified destination.

#### Parameters

<i>host</i>	The host name or IP address of the remote host to connect to.
<i>port</i>	The port on the remote host to connect to.

#### Exceptions

<i>IOException</i>	Thrown if a failure occurred in the connect.
<i>IllegalArgumentEx-ception</i>	if the timeout value is negative or the endpoint is invalid.

**6.746.3.7** `void decaf::net::Socket::ensureCreated ( ) const throw ( decaf::io::IOException )`  
`[protected]`

**6.746.3.8** `std::string decaf::net::Socket::getInetAddress ( ) const`

Returns the address to which the socket is connected.

#### Returns

the remote IP address to which this socket is connected, or null if the socket is not connected.

**6.746.3.9** `virtual decaf::io::InputStream* decaf::net::Socket::getInputStream ( ) throw ( decaf::io::IOException )` `[virtual]`

Gets the InputStream for this socket if its connected.

The pointer returned is the property of the associated **Socket** (p. 3607) and should not be deleted by the caller.

When the returned InputStream is performing a blocking operation and the underlying connection is closed or otherwise broker the read calls will normally throw an exception to indicate the failure.

Closing the InputStream will also close the underlying **Socket** (p. 3607).

#### Returns

The InputStream for this socket.

#### Exceptions

<i>IOException</i>	if an error occurs during creation of the InputStream, also if the <b>Socket</b> (p. 3607) is not connected or the input has been shutdown previously.
--------------------	--

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2948).

**6.746.3.10** `virtual bool decaf::net::Socket::getKeepAlive ( ) const throw ( SocketException )`  
`[virtual]`

Gets the keep alive flag for this socket, SO\_KEEPALIVE.

#### Returns

true if keep alive is enabled for this socket.

#### Exceptions

<i>SocketException</i> (p. 3627)	if the operation fails.
-------------------------------------	-------------------------

**6.746.3.11** `std::string decaf::net::Socket::getLocalAddress ( ) const`

Gets the local address to which the socket is bound.

**Returns**

the local address to which the socket is bound or `InetAddress.anyLocalAddress()` if the socket is not bound yet.

**6.746.3.12** `int decaf::net::Socket::getLocalPort ( ) const`

Gets the local port the socket is bound to.

**Returns**

the local port the socket was bound to, or -1 if the socket is not bound.

**6.746.3.13** `virtual bool decaf::net::Socket::getOOBInline ( ) const throw ( SocketException )`  
[virtual]

Gets the value of the OOBINLINE for this socket.

**Returns**

true if OOBINLINE is enabled, false otherwise.

**Exceptions**

<i>SocketException</i> (p. 3627)	if an error is encountered while performing this operation.
-------------------------------------	---

**6.746.3.14** `virtual decaf::io::OutputStream* decaf::net::Socket::getOutputStream ( )`  
`throw ( decaf::io::IOException )` [virtual]

Gets the OutputStream for this socket if it is connected.

The pointer returned is the property of the **Socket** (p. 3607) instance and should not be deleted by the caller.

Closing the returned **Socket** (p. 3607) will also close the underlying **Socket** (p. 3607).

**Returns**

the OutputStream for this socket.

**Exceptions**

<i>IOException</i>	if an error occurs during the creation of this OutputStream, or if the <b>Socket</b> (p. 3607) is closed or the output has been shutdown previously.
--------------------	--

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2949).

**6.746.3.15** `int decaf::net::Socket::getPort ( ) const`

Gets the on the remote host this **Socket** (p. 3607) is connected to.

#### Returns

the port on the remote host the socket is connected to, or 0 if not connected.

**6.746.3.16** `virtual int decaf::net::Socket::getReceiveBufferSize ( ) const throw ( SocketException ) [virtual]`

Gets the receive buffer size for this socket, SO\_RCVBUF.

This is the buffer used by the underlying platform socket to buffer received data.

#### Returns

the receive buffer size in bytes.

#### Exceptions

<i><b>SocketException</b></i> (p. 3627)	if the operation fails.
--	-------------------------

**6.746.3.17** `virtual bool decaf::net::Socket::getReuseAddress ( ) const throw ( SocketException ) [virtual]`

Gets the reuse address flag, SO\_REUSEADDR.

#### Returns

True if the address can be reused.

#### Exceptions

<i><b>SocketException</b></i> (p. 3627)	if the operation fails.
--	-------------------------

**6.746.3.18** `virtual int decaf::net::Socket::getSendBufferSize ( ) const throw ( SocketException ) [virtual]`

Gets the send buffer size for this socket, SO\_SNDBUF, this value is used by the platform socket to buffer data written to the socket.

**Returns**

the size in bytes of the send buffer.

**Exceptions**

<i>SocketException</i> (p. 3627)	if the operation fails.
-------------------------------------	-------------------------

**6.746.3.19** `virtual int decaf::net::Socket::getSoLinger ( ) const throw ( SocketException )`  
[virtual]

Gets the linger time for the socket, SO\_LINGER.

A return value of -1 indicates that the option is disabled.

**Returns**

The linger time in seconds.

**Exceptions**

<i>SocketException</i> (p. 3627)	if the operation fails.
-------------------------------------	-------------------------

**6.746.3.20** `virtual int decaf::net::Socket::getSoTimeout ( ) const throw ( SocketException )`  
[virtual]

Gets the timeout for socket operations, SO\_TIMEOUT.

**Returns**

The timeout in milliseconds for socket operations.

**Exceptions**

<i>SocketException</i> (p. 3627)	Thrown if unable to retrieve the information.
-------------------------------------	---

**6.746.3.21** `virtual bool decaf::net::Socket::getTcpNoDelay ( ) const throw ( SocketException )`  
[virtual]

Gets the Status of the TCP\_NODELAY setting for this socket.

**Returns**

true if TCP\_NODELAY is enabled for the socket.

**Exceptions**

<b><i>SocketException</i></b> (p. 3627)	Thrown if unable to set the information.
--	--

**6.746.3.22** `virtual int decaf::net::Socket::getTrafficClass ( ) const throw ( SocketException )`  
[virtual]

Gets the Traffic Class setting for this **Socket** (p. 3607), sometimes referred to as Type of Service setting.

This setting is dependent on the underlying network implementation for the platform this **Socket** (p. 3607) runs on and is not guaranteed to have any effect.

Refer to your platforms network documentation regarding support for this setting.

**Returns**

the bitset result of querying the traffic class setting.

**Exceptions**

<b><i>SocketException</i></b> (p. 3627)	if an error is encountered while performing this operation.
--	---

**6.746.3.23** `void decaf::net::Socket::initSocketImpl ( const std::string & address, int port, const InetAddress * localAddress, int localPort ) throw ( decaf::io::IOException, decaf::net::UnknownHostException )` [protected]

**6.746.3.24** `bool decaf::net::Socket::isBound ( ) const` [inline]

**Returns**

true if this **Socket** (p. 3607) has been bound to a Local address.

**6.746.3.25** `bool decaf::net::Socket::isClosed ( ) const` [inline]

**Returns**

true if the **Socket** (p. 3607) has been closed.

**6.746.3.26** `bool decaf::net::Socket::isConnected ( ) const` [inline]

Indicates whether or not this socket is connected to an end point.

**Returns**

true if connected, false otherwise.



**6.746.3.27** `bool decaf::net::Socket::isInputShutdown ( ) const [inline]`

#### Returns

true if input on this **Socket** (p. 3607) has been shutdown.

**6.746.3.28** `bool decaf::net::Socket::isOutputShutdown ( ) const [inline]`

#### Returns

true if output on this **Socket** (p. 3607) has been shutdown.

**6.746.3.29** `virtual void decaf::net::Socket::sendUrgentData ( int data ) throw ( decaf::io::IOException ) [virtual]`

Sends on byte of urgent data to the **Socket** (p. 3607).

#### Parameters

<i>data</i>	The value to write as urgent data, only the lower eight bits are sent.
-------------	--

#### Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2951).

**6.746.3.30** `virtual void decaf::net::Socket::setKeepAlive ( bool keepAlive ) throw ( SocketException ) [virtual]`

Enables/disables the keep alive flag for this socket, SO\_KEEPALIVE.

#### Parameters

<i>keepAlive</i>	If true, enables the flag.
------------------	----------------------------

#### Exceptions

<i>SocketException</i> (p. 3627)	if the operation fails.
-------------------------------------	-------------------------

**6.746.3.31** `virtual void decaf::net::Socket::setOOBInline ( bool value ) throw ( SocketException ) [virtual]`

Sets the value of the OOBINLINE for this socket, by default this option is disabled.

If enabled the urgent data is read inline on the Socket's InputStream, no notification is give.

**Returns**

true if OOBINLINE is enabled, false otherwise.

**Exceptions**

<i><b>SocketException</b></i> (p. 3627)	if an error is encountered while performing this operation.
--	---

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2952).

**6.746.3.32** `virtual void decaf::net::Socket::setReceiveBufferSize ( int size ) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )`  
[virtual]

Sets the receive buffer size for this socket, SO\_RCVBUF.

**Parameters**

<i>size</i>	Number of bytes to set the receive buffer to.
-------------	---

**Exceptions**

<i><b>SocketException</b></i> (p. 3627)	if the operation fails.
<i><b>IllegalArgumentException</b></i>	if the value is zero or negative.

**6.746.3.33** `virtual void decaf::net::Socket::setReuseAddress ( bool reuse ) throw ( SocketException )` [virtual]

Sets the reuse address flag, SO\_REUSEADDR.

**Parameters**

<i>reuse</i>	If true, sets the flag.
--------------	-------------------------

**Exceptions**

<i><b>SocketException</b></i> (p. 3627)	if the operation fails.
--	-------------------------

**6.746.3.34** `virtual void decaf::net::Socket::setSendBufferSize ( int size ) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )`  
[virtual]

Gets the send buffer size for this socket, SO\_SNDBUF, this value is used by the platform socket to buffer data written to the socket.

**Parameters**

<i>size</i>	The number of bytes to set the send buffer to, must be larger than zero.
-------------	--

**Exceptions**

<b><i>SocketException</i></b> (p. 3627)	if the operation fails.
<b><i>IllegalArgumentException</i></b>	if the value is zero or negative.

**6.746.3.35** `static void decaf::net::Socket::setSocketImplFactory ( SocketImplFactory *  
factory ) throw ( decaf::io::IOException, decaf::net::SocketException )  
[static]`

Sets the instance of a **SocketImplFactory** (p. 3644) that the **Socket** (p. 3607) class should use when new instances of this class are created.

This method is only allowed to be used once during the lifetime of the application.

**Parameters**

<i>factory</i>	The instance of a <b>SocketImplFactory</b> (p. 3644) to use when new <b>Socket</b> (p. 3607) objects are created.
----------------	---

**Exceptions**

<b><i>IOException</i></b>	if an I/O error occurs while performing this operation.
<b><i>SocketException</i></b> (p. 3627)	if this method has already been called with a valid factory.

**6.746.3.36** `virtual void decaf::net::Socket::setSoLinger ( bool state, int timeout ) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )  
[virtual]`

Sets the linger time (SO\_LINGER) using a specified time value, this limits of this value are platform specific.

**Parameters**

<i>state</i>	The state of SO_LINGER, true is on.
<i>timeout</i>	The linger time in seconds, must be non-negative.

**Exceptions**

<b><i>SocketException</i></b> (p. 3627)	if the operation fails.
<b><i>IllegalArgumentException</i></b>	if state is true and timeout is negative.

**6.746.3.37** `virtual void decaf::net::Socket::setSoTimeout ( int timeout ) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )`  
[virtual]

Sets the timeout for socket operations, SO\_TIMEOUT.

A value of zero indicates that timeout is infinite for operations on this socket.

#### Parameters

<i>timeout</i>	The timeout in milliseconds for socket operations.
----------------	--

#### Exceptions

<b>SocketException</b> (p. 3627)	Thrown if unable to set the information.
<b>IllegalArgumentException</b>	if the timeout value is negative.

**6.746.3.38** `virtual void decaf::net::Socket::setTcpNoDelay ( bool value ) throw ( SocketException )` [virtual]

Sets the Status of the TCP\_NODELAY param for this socket., this setting is used to disable or enable Nagle's algorithm on the **Socket** (p. 3607).

#### Parameters

<i>value</i>	The setting for the socket's TCP_NODELAY option, true to enable.
--------------	--

#### Exceptions

<b>SocketException</b> (p. 3627)	Thrown if unable to set the information.
-------------------------------------	--

**6.746.3.39** `virtual void decaf::net::Socket::setTrafficClass ( int value ) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )`  
[virtual]

Gets the Traffic Class setting for this **Socket** (p. 3607), sometimes referred to as Type of Service setting.

This setting is dependent on the underlying network implementation for the platform this **Socket** (p. 3607) runs on and is not guaranteed to have any effect.

Refer to your platforms network documentation regarding support for this setting.

#### Parameters

<i>value</i>	The integer value representing the traffic class setting bitset.
--------------	--

**Exceptions**

<i>SocketException</i> (p. 3627)	if an error is encountered while performing this operation.
<i>IllegalArgumentEx- ception</i>	if the value is not in the range [0..255].

**6.746.3.40** virtual void decaf::net::Socket::shutdownInput ( ) throw ( decaf::io::IOException ) [virtual]

Shuts down the InputStream for this socket essentially marking it as EOF.

The stream returns EOF for any calls to read after this method has been called.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2953).

**6.746.3.41** virtual void decaf::net::Socket::shutdownOutput ( ) throw ( decaf::io::IOException ) [virtual]

Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to OutputStream::write will throw an IOException.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2954).

**6.746.3.42** virtual std::string decaf::net::Socket::toString ( ) const [virtual]

**Returns**

a string representing this **Socket** (p. 3607).

**6.746.4 Friends And Related Function Documentation**

**6.746.4.1** friend class ServerSocket [friend]

**6.746.5 Field Documentation**

**6.746.5.1** SocketImpl\* decaf::net::Socket::impl [mutable, protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/net/Socket.h`

## 6.747 decaf::net::SocketAddress Class Reference

Base class for protocol specific **Socket** (p. 3607) addresses.

```
#include <src/main/decaf/net/SocketAddress.h>
```

Inheritance diagram for `decaf::net::SocketAddress`:

### Public Member Functions

- `virtual ~SocketAddress ()`

#### 6.747.1 Detailed Description

Base class for protocol specific **Socket** (p. 3607) addresses. These classes provide an immutable address object that is used by the **Socket** (p. 3607) classes.

Since

1.0

#### 6.747.2 Constructor & Destructor Documentation

6.747.2.1 `virtual decaf::net::SocketAddress::~~SocketAddress ( ) [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketAddress.h`

## 6.748 decaf::net::SocketError Class Reference

Static utility class to simplify handling of error codes for socket operations.

```
#include <src/main/decaf/net/SocketError.h>
```

### Static Public Member Functions

- `static int getErrorCode ()`  
*Gets the last error appropriate for the platform.*

- static std::string **getErrorString** ()  
*Gets the string description for the last error.*

### 6.748.1 Detailed Description

Static utility class to simplify handling of error codes for socket operations.

### 6.748.2 Member Function Documentation

#### 6.748.2.1 static int decaf::net::SocketError::getErrorCode ( ) [static]

Gets the last error appropriate for the platform.

#### 6.748.2.2 static std::string decaf::net::SocketError::getErrorString ( ) [static]

Gets the string description for the last error.

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketError.h**

## 6.749 decaf::net::SocketException Class Reference

Exception for errors when manipulating sockets.

```
#include <src/main/decaf/net/SocketException.h>
```

Inheritance diagram for decaf::net::SocketException:

### Public Member Functions

- **SocketException** () throw ()
- **SocketException** (const lang::Exception &ex) throw ()
- **SocketException** (const **SocketException** &ex) throw ()
- **SocketException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **SocketException** (const std::exception \*cause) throw ()  
*Constructor.*
- **SocketException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()

*Constructor - Initializes the file name and line number where this message occurred.*

- virtual **SocketException** \* **clone** () const  
*Clones this exception.*
- virtual ~**SocketException** () throw ()

### 6.749.1 Detailed Description

Exception for errors when manipulating sockets.

### 6.749.2 Constructor & Destructor Documentation

6.749.2.1 **decaf::net::SocketException::SocketException** ( ) throw () [inline]

6.749.2.2 **decaf::net::SocketException::SocketException** ( const lang::Exception & *ex* ) throw () [inline]

6.749.2.3 **decaf::net::SocketException::SocketException** ( const SocketException & *ex* ) throw () [inline]

6.749.2.4 **decaf::net::SocketException::SocketException** ( const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.749.2.5 **decaf::net::SocketException::SocketException** ( const std::exception \* *cause* ) throw () [inline]

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--



**6.749.2.6** `decaf::net::SocketException::SocketException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.749.2.7** `virtual decaf::net::SocketException::~~SocketException ( ) throw () [inline, virtual]`

### 6.749.3 Member Function Documentation

**6.749.3.1** `virtual SocketException* decaf::net::SocketException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 2212).

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketException** (p. 2958), **decaf::net::BindException** (p. 844), **decaf::net::ConnectException** (p. 1296), **decaf::net::NoRouteToHostException** (p. 2907), and **decaf::net::PortUnreachableException** (p. 3062).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketException.h`

## 6.750 decaf::net::SocketFactory Class Reference

The **SocketFactory** (p. 3629) is used to create **Socket** (p. 3607) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.

```
#include <src/main/decaf/net/SocketFactory.h>
```

Inheritance diagram for decaf::net::SocketFactory:

## Public Member Functions

- virtual **~SocketFactory** ()
- virtual **Socket \* createSocket** () throw ( decaf::io::IOException )  
*Creates an unconnected **Socket** (p. 3607) object.*
- virtual **Socket \* createSocket** (const **InetAddress** \*host, int port)=0 throw ( decaf::io::IOException, decaf::net::UnknownHostException )  
*Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).*
- virtual **Socket \* createSocket** (const **InetAddress** \*host, int port, const **InetAddress** \*ifAddress, int localPort)=0 throw ( decaf::io::IOException, decaf::net::UnknownHostException )  
*Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).*
- virtual **Socket \* createSocket** (const std::string &name, int port)=0 throw ( decaf::io::IOException, decaf::net::UnknownHostException )  
*Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).*
- virtual **Socket \* createSocket** (const std::string &name, int port, const **InetAddress** \*ifAddress, int localPort)=0 throw ( decaf::io::IOException, decaf::net::UnknownHostException )  
*Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).*

## Static Public Member Functions

- static **SocketFactory \* getDefault** ()  
*Returns an pointer to the default **SocketFactory** (p. 3629) for this Application, there is only one default **SocketFactory** (p. 3629) per application, the pointer returned by this method is owned by the **SocketFactory** (p. 3629) class and in not to be deleted by the caller.*

## Protected Member Functions

- **SocketFactory** ()

### 6.750.1 Detailed Description

The **SocketFactory** (p. 3629) is used to create **Socket** (p. 3607) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.

#### See also

**decaf.net.Socket** (p. 3607)

#### Since

1.0

### 6.750.2 Constructor & Destructor Documentation

6.750.2.1 **decaf::net::SocketFactory::SocketFactory** ( ) [protected]

6.750.2.2 **virtual decaf::net::SocketFactory::~~SocketFactory** ( ) [virtual]

### 6.750.3 Member Function Documentation

6.750.3.1 **virtual Socket\*** **decaf::net::SocketFactory::createSocket** ( ) throw ( **decaf::io::IOException** ) [virtual]

Creates an unconnected **Socket** (p. 3607) object.

#### Returns

a new **Socket** (p. 3607) object, caller must free this object when done.

#### Exceptions

<i>IOException</i>	if the <b>Socket</b> (p. 3607) cannot be created.
--------------------	---

Reimplemented in **decaf::internal::net::DefaultSocketFactory** (p. 1741), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1753), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2962).

6.750.3.2 **virtual Socket\*** **decaf::net::SocketFactory::createSocket** ( **const InetAddress \*** *host*, **int** *port* ) throw ( **decaf::io::IOException**, **decaf::net::UnknownHostException** ) [pure virtual]

Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).

#### Parameters

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

**Returns**

a new **Socket** (p. 3607) object, caller must free this object when done.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<b>UnknownHostException</b> (p. 4019)	if the host name is not known.

Implemented in **decaf::internal::net::DefaultSocketFactory** (p. 1743), **decaf::internal::net::ssl::DefaultSSL** (p. 1754), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2963).

```
6.750.3.3  virtual Socket* decaf::net::SocketFactory::createSocket ( const std::string
& name, int port, const InetAddress * ifAddress, int localPort ) throw (
decaf::io::IOException, decaf::net::UnknownHostException ) [pure
virtual]
```

Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).

**Parameters**

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the <b>Socket</b> (p. 3607) to.
<i>localPort</i>	The local port to bind the <b>Socket</b> (p. 3607) to.

**Returns**

a new **Socket** (p. 3607) object, caller must free this object when done.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<b>UnknownHostException</b> (p. 4019)	if the host name is not known.

Implemented in **decaf::internal::net::DefaultSocketFactory** (p. 1741), **decaf::internal::net::ssl::DefaultSSL** (p. 1755), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2964).

```
6.750.3.4  virtual Socket* decaf::net::SocketFactory::createSocket ( const
std::string & name, int port ) throw ( decaf::io::IOException,
decaf::net::UnknownHostException ) [pure virtual]
```

Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).

**Parameters**

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

**Returns**

a new **Socket** (p. 3607) object, caller must free this object when done.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<b>UnknownHostException</b> (p. 4019)	if the host name is not known.

Implemented in **decaf::internal::net::DefaultSocketFactory** (p. 1742), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1755), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2964).

```
6.750.3.5 virtual Socket* decaf::net::SocketFactory::createSocket ( const InetAddress
* host, int port, const InetAddress * ifAddress, int localPort ) throw (
decaf::io::IOException, decaf::net::UnknownHostException ) [pure
virtual]
```

Creates a new **Socket** (p. 3607) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3629).

The **Socket** (p. 3607) will be bound to the specified local address and port.

**Parameters**

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the <b>Socket</b> (p. 3607) to.
<i>localPort</i>	The local port to bind the <b>Socket</b> (p. 3607) to.

**Returns**

a new **Socket** (p. 3607) object, caller must free this object when done.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3607) object.
<b>UnknownHostException</b> (p. 4019)	if the host name is not known.

Implemented in **decaf::internal::net::DefaultSocketFactory** (p. 1742), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1756), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2965).

**6.750.3.6 static SocketFactory\* decaf::net::SocketFactory::getDefault ( ) [static]**

Returns an pointer to the default **SocketFactory** (p. 3629) for this Application, there is only one default **SocketFactory** (p. 3629) per application, the pointer returned by this method is owned by the **SocketFactory** (p. 3629) class and in not to be deleted by the caller.

**Returns**

pointer to the applications default **SocketFactory** (p. 3629).

**Exceptions**

<b>SocketException</b> (p. 3627)	if an error occurs while getting the default instance.
-------------------------------------	--

Reimplemented in **decaf::net::ssl::SSLSocketFactory** (p. 3681).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketFactory.h**

**6.751 decaf::internal::net::SocketFileDescriptor Class Reference**

File Descriptor type used internally by Decaf Socket objects.

```
#include <src/main/decaf/internal/net/SocketFileDescriptor.h>
```

Inheritance diagram for decaf::internal::net::SocketFileDescriptor:

**Public Member Functions**

- **SocketFileDescriptor** (long value)
- virtual **~SocketFileDescriptor** ()
- long **getValue** () const

*Gets the OS Level FileDescriptor.*

**6.751.1 Detailed Description**

File Descriptor type used internally by Decaf Socket objects.

**Since**

1.0

## 6.751.2 Constructor & Destructor Documentation

6.751.2.1 decaf::internal::net::SocketFileDescriptor::SocketFileDescriptor ( long *value* )

6.751.2.2 virtual decaf::internal::net::SocketFileDescriptor::~~SocketFileDescriptor ( )  
[virtual]

## 6.751.3 Member Function Documentation

6.751.3.1 long decaf::internal::net::SocketFileDescriptor::getValue ( ) const

Gets the OS Level FileDescriptor.

### Returns

a FileDescriptor value.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/SocketFileDescriptor.h

## 6.752 decaf::net::SocketImpl Class Reference

Acts as a base class for all physical **Socket** (p. 3607) implementations.

```
#include <src/main/decaf/net/SocketImpl.h>
```

Inheritance diagram for decaf::net::SocketImpl:

### Public Member Functions

- **SocketImpl** ()
- virtual ~**SocketImpl** ()
- virtual void **create** ()=0 throw ( decaf::io::IOException )  
*Creates the underlying platform **Socket** (p.3607) data structures which allows for **Socket** (p. 3607) options to be applied.*
- virtual void **accept** (**SocketImpl** \*socket)=0 throw ( decaf::io::IOException, decaf::net::SocketException, decaf::net::SocketTimeoutException )  
*Accepts a new connection on the given **Socket** (p. 3607).*
- virtual void **connect** (const std::string &hostname, int **port**, int timeout)=0 throw ( decaf::io::IOException, decaf::net::SocketTimeoutException, decaf::lang::exceptions::IllegalArgumentException )  
*Connects this socket to the given host and port.*

- virtual void **bind** (const std::string &ipaddress, int **port**)=0 throw ( decaf::io::IOException )  
*Binds this **Socket** (p. 3607) instance to the local ip address and port number given.*
- virtual void **listen** (int backlog)=0 throw ( decaf::io::IOException )  
*Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.*
- virtual **decaf::io::InputStream** \* **getInputStream** ()=0 throw ( decaf::io::IOException )  
*Gets the **InputStream** linked to this **Socket** (p. 3607).*
- virtual **decaf::io::OutputStream** \* **getOutputStream** ()=0 throw ( decaf::io::IOException )  
*Gets the **OutputStream** linked to this **Socket** (p. 3607).*
- virtual int **available** ()=0 throw ( decaf::io::IOException )  
*Gets the number of bytes that can be read from the **Socket** (p. 3607) without blocking.*
- virtual void **close** ()=0 throw ( decaf::io::IOException )  
*Closes the socket, terminating any blocked reads or writes.*
- virtual void **shutdownInput** ()=0 throw ( decaf::io::IOException )  
*Places the input stream for this socket at "end of stream".*
- virtual void **shutdownOutput** ()=0 throw ( decaf::io::IOException )  
*Disables the output stream for this socket.*
- virtual int **getOption** (int option) const =0 throw ( decaf::io::IOException )  
*Gets the specified **Socket** (p. 3607) option.*
- virtual void **setOption** (int option, int value)=0 throw ( decaf::io::IOException )  
*Sets the specified option on the **Socket** (p. 3607) if supported.*
- int **getPort** () const  
*Gets the port that this socket has been assigned.*
- int **getLocalPort** () const  
*Gets the value of this **SocketImpl**'s local port field.*
- std::string **getInetAddress** () const  
*Gets the value of this **SocketImpl**'s address field.*
- const **decaf::io::FileDescriptor** \* **getFileDescriptor** () const



*Gets the FileDescriptor for this **Socket** (p. 3607), the Object is owned by this **Socket** (p. 3607) and should not be deleted by the caller.*

- virtual std::string **getLocalAddress** () const =0

*Gets the value of the local Inet address the **Socket** (p. 3607) is bound to if bound, otherwise return the **InetAddress** (p. 2077) ANY value "0.0.0.0".*

- std::string **toString** () const

*Returns a string containing the address and port of this **Socket** (p. 3607) instance.*

- virtual bool **supportsUrgentData** () const
- virtual void **sendUrgentData** (int data) throw ( decaf::io::IOException )

*Sends on byte of urgent data to the **Socket** (p. 3607).*

## Protected Attributes

- int **port**

*The remote port that this **Socket** (p. 3607) is connected to.*

- int **localPort**

*The port on the Local Machine that this **Socket** (p. 3607) is Bound to.*

- std::string **address**

*The Remote Address that the **Socket** (p. 3607) is connected to.*

- io::FileDescriptor \* **fd**

*The File Descriptor for this **Socket** (p. 3607).*

### 6.752.1 Detailed Description

Acts as a base class for all physical **Socket** (p. 3607) implementations.

Since

1.0

## 6.752.2 Constructor & Destructor Documentation

6.752.2.1 `decaf::net::SocketImpl::SocketImpl ( )`

6.752.2.2 `virtual decaf::net::SocketImpl::~~SocketImpl ( )` [virtual]

## 6.752.3 Member Function Documentation

6.752.3.1 `virtual void decaf::net::SocketImpl::accept ( SocketImpl * socket )  
throw ( decaf::io::IOException, decaf::net::SocketException,  
decaf::net::SocketTimeoutException )` [pure virtual]

Accepts a new connection on the given **Socket** (p. 3607).

### Parameters

<i>socket</i>	The accepted connection.
---------------	--------------------------

### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
<i>SocketException</i> (p. 3627)	if an error occurs while performing an Accept on the socket.
<i>SocketTimeoutException</i> (p. 3650)	if the accept call times out due to SO_TIMEOUT being set.

6.752.3.2 `virtual int decaf::net::SocketImpl::available ( ) throw ( decaf::io::IOException )`  
[pure virtual]

Gets the number of bytes that can be read from the **Socket** (p. 3607) without blocking.

### Returns

the number of bytes that can be read from the **Socket** (p. 3607) without blocking.

### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3854).

6.752.3.3 `virtual void decaf::net::SocketImpl::bind ( const std::string & ipaddress, int port )  
throw ( decaf::io::IOException )` [pure virtual]

Binds this **Socket** (p. 3607) instance to the local ip address and port number given.

**Parameters**

<i>ipaddress</i>	The address of local ip to bind to.
<i>port</i>	The port number on the host to bind to.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3854).

**6.752.3.4** `virtual void decaf::net::SocketImpl::close ( ) throw ( decaf::io::IOException )`  
[pure virtual]

Closes the socket, terminating any blocked reads or writes.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3855).

**6.752.3.5** `virtual void decaf::net::SocketImpl::connect ( const  
std::string & hostname, int port, int timeout ) throw (`  
**decaf::io::IOException, decaf::net::SocketTimeoutException,**  
**decaf::lang::exceptions::IllegalArgumentException )** [pure  
virtual]

Connects this socket to the given host and port.

**Parameters**

<i>hostname</i>	The name of the host to connect to, or IP address.
<i>port</i>	The port number on the host to connect to.
<i>timeout</i>	Time in milliseconds to wait for a connection, 0 indicates forever.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while attempting this operation.
<b>SocketTimeoutException</b> (p. 3650)	if the connect call times out due to timeout being set.
<i>IllegalArgumentEx- ception</i>	if a parameter has an illegal value.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3855).

**6.752.3.6** `virtual void decaf::net::SocketImpl::create ( ) throw ( decaf::io::IOException )`  
`[pure virtual]`

Creates the underlying platform **Socket** (p. 3607) data structures which allows for **Socket** (p. 3607) options to be applied.

The created socket is in an unconnected state.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3855).

**6.752.3.7** `const decaf::io::FileDescriptor* decaf::net::SocketImpl::getFileDescriptor ( )`  
`const [inline]`

Gets the FileDescriptor for this **Socket** (p. 3607), the Object is owned by this **Socket** (p. 3607) and should not be deleted by the caller.

#### Returns

a pointer to this Socket's FileDescriptor object.

**6.752.3.8** `std::string decaf::net::SocketImpl::getInetAddress ( ) const [inline]`

Gets the value of this SocketImpl's address field.

#### Returns

the value of the address field.

**6.752.3.9** `virtual decaf::io::InputStream* decaf::net::SocketImpl::getInputStream ( ) throw ( decaf::io::IOException )`  
`[pure virtual]`

Gets the InputStream linked to this **Socket** (p. 3607).

#### Returns

an InputStream pointer owned by the **Socket** (p. 3607) object.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3856).

**6.752.3.10** `virtual std::string decaf::net::SocketImpl::getLocalAddress ( ) const` [pure virtual]

Gets the value of the local Inet address the **Socket** (p. 3607) is bound to if bound, otherwise return the **InetAddress** (p. 2077) ANY value "0.0.0.0".

#### Returns

the local address bound to, or ANY.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3856).

**6.752.3.11** `int decaf::net::SocketImpl::getLocalPort ( ) const` [inline]

Gets the value of this SocketImpl's local port field.

#### Returns

the value of localPort.

**6.752.3.12** `virtual int decaf::net::SocketImpl::getOption ( int option ) const throw ( decaf::io::IOException )` [pure virtual]

Gets the specified **Socket** (p. 3607) option.

#### Parameters

<i>option</i>	The <b>Socket</b> (p. 3607) options whose value is to be retrieved.
---------------	---

#### Returns

the value of the given socket option.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3856).

**6.752.3.13** `virtual decaf::io::OutputStream* decaf::net::SocketImpl::getOutputStream ( ) throw ( decaf::io::IOException )` [pure virtual]

Gets the OutputStream linked to this **Socket** (p. 3607).

#### Returns

an OutputStream pointer owned by the **Socket** (p. 3607) object.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3857).

**6.752.3.14** `int decaf::net::SocketImpl::getPort ( ) const` `[inline]`

Gets the port that this socket has been assigned.

**Returns**

the Socket's port number.

**6.752.3.15** `virtual void decaf::net::SocketImpl::listen ( int backlog ) throw ( decaf::io::IOException )` `[pure virtual]`

Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.

If a connection indication arrives when the queue is full, the connection is refused.

**Parameters**

<i>backlog</i>	The maximum length of the connection queue.
----------------	---

**Exceptions**

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3857).

**6.752.3.16** `virtual void decaf::net::SocketImpl::sendUrgentData ( int data ) throw ( decaf::io::IOException )` `[virtual]`

Sends on byte of urgent data to the **Socket** (p. 3607).

**Parameters**

<i>data</i>	The value to write as urgent data, only the lower eight bits are sent.
-------------	--

**Exceptions**

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

**6.752.3.17** virtual void decaf::net::SocketImpl::setOption ( int *option*, int *value* ) throw ( decaf::io::IOException ) [pure virtual]

Sets the specified option on the **Socket** (p. 3607) if supported.

#### Parameters

<i>option</i>	The <b>Socket</b> (p. 3607) option to set.
<i>value</i>	The value of the socket option to apply to the socket.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3858).

**6.752.3.18** virtual void decaf::net::SocketImpl::shutdownInput ( ) throw ( decaf::io::IOException ) [pure virtual]

Places the input stream for this socket at "end of stream".

Any data sent to this socket is acknowledged and then silently discarded. If you read from a socket input stream after invoking **shutdownInput()** (p. 3643) on the socket, the stream will return EOF.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3859).

**6.752.3.19** virtual void decaf::net::SocketImpl::shutdownOutput ( ) throw ( decaf::io::IOException ) [pure virtual]

Disables the output stream for this socket.

For a TCP socket, any previously written data will be sent followed by TCP's normal connection termination sequence. If you write to a socket output stream after invoking **shutdownOutput()** (p. 3643) on the socket, the stream will throw an IOException.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3859).

**6.752.3.20** `virtual bool decaf::net::SocketImpl::supportsUrgentData ( ) const` `[inline, virtual]`

#### Returns

true if this **SocketImpl** (p. 3635) supports sending Urgent Data. The default implementation always returns false.

**6.752.3.21** `std::string decaf::net::SocketImpl::toString ( ) const`

Returns a string containing the address and port of this **Socket** (p. 3607) instance.

#### Returns

a string containing the address and port of this socket.

### 6.752.4 Field Documentation

**6.752.4.1** `std::string decaf::net::SocketImpl::address` `[protected]`

The Remote Address that the **Socket** (p. 3607) is connected to.

**6.752.4.2** `io::FileDescriptor* decaf::net::SocketImpl::fd` `[protected]`

The File Descriptor for this **Socket** (p. 3607).

**6.752.4.3** `int decaf::net::SocketImpl::localPort` `[protected]`

The port on the Local Machine that this **Socket** (p. 3607) is Bound to.

**6.752.4.4** `int decaf::net::SocketImpl::port` `[protected]`

The remote port that this **Socket** (p. 3607) is connected to.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketImpl.h`

### 6.753 decaf::net::SocketImplFactory Class Reference

Factory class interface for a Factory that creates SocketImpl objects.

```
#include <src/main/decaf/net/SocketImplFactory.h>
```



## Public Member Functions

- virtual `~SocketImplFactory()`
- virtual `SocketImpl * createSocketImpl()=0`

*Creates a new SocketImpl instance and returns it, the caller then owns the instance and must delete it when finished with the **SocketImpl** (p. 3635).*

### 6.753.1 Detailed Description

Factory class interface for a Factory that creates SocketImpl objects. These factories can be used to create various types of Sockets, e.g. Streaming, Multicast, SSL, or platform specific variations of these types.

#### See also

**decaf::net::Socket** (p. 3607)  
**decaf::net::ServerSocket** (p. 3447)

#### Since

1.0

### 6.753.2 Constructor & Destructor Documentation

**6.753.2.1** virtual `decaf::net::SocketImplFactory::~~SocketImplFactory()` [`inline`, `virtual`]

### 6.753.3 Member Function Documentation

**6.753.3.1** virtual `SocketImpl* decaf::net::SocketImplFactory::createSocketImpl()` [`pure` `virtual`]

Creates a new SocketImpl instance and returns it, the caller then owns the instance and must delete it when finished with the **SocketImpl** (p. 3635).

#### Returns

new **SocketImpl** (p. 3635) instance that is owned by the caller.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketImplFactory.h`

## 6.754 decaf::net::SocketOptions Class Reference

```
#include <src/main/decaf/net/SocketOptions.h>
```

Inheritance diagram for decaf::net::SocketOptions:

## Public Member Functions

- virtual  $\sim$ SocketOptions ()

## Static Public Attributes

- static const int **SOCKET\_OPTION\_TCP\_NODELAY**  
*Disable Nagle's algorithm for this connection.*
- static const int **SOCKET\_OPTION\_BINDADDR**  
*Fetch the local address binding of a socket (this option cannot be "set" only "gotten", since sockets are bound at creation time, and so the locally bound address cannot be changed).*
- static const int **SOCKET\_OPTION\_REUSEADDR**  
*Sets SO\_REUSEADDR for a socket.*
- static const int **SOCKET\_OPTION\_BROADCAST**  
*Sets SO\_BROADCAST for a socket.*
- static const int **SOCKET\_OPTION\_IP\_MULTICAST\_IF**  
*Set which outgoing interface on which to send multicast packets.*
- static const int **SOCKET\_OPTION\_IP\_MULTICAST\_IF2**  
*Same as above.*
- static const int **SOCKET\_OPTION\_IP\_MULTICAST\_LOOP**  
*This option enables or disables local loopback of multicast datagrams.*
- static const int **SOCKET\_OPTION\_IP\_TOS**  
*This option sets the type-of-service or traffic class field in the IP header for a TCP or UDP socket.*
- static const int **SOCKET\_OPTION\_LINGER**  
*Specify a linger-on-close timeout.*
- static const int **SOCKET\_OPTION\_TIMEOUT**  
*Set a timeout on blocking **Socket** (p. 3607) operations.*
- static const int **SOCKET\_OPTION\_SNDBUF**  
*Set a hint the size of the underlying buffers used by the platform for outgoing network I/O.*

- static const int **SOCKET\_OPTION\_RCVBUF**

*Set a hint the size of the underlying buffers used by the platform for incoming network I/O.*

- static const int **SOCKET\_OPTION\_KEEPALIVE**

*When the keepalive option is set for a TCP socket and no data has been exchanged across the socket in either direction for 2 hours (NOTE: the actual value is implementation dependent), TCP automatically sends a keepalive probe to the peer.*

- static const int **SOCKET\_OPTION\_OOINLINE**

*When the OOBINLINE option is set, any TCP urgent data received on the socket will be received through the socket input stream.*

### 6.754.1 Detailed Description

Since

1.0

### 6.754.2 Constructor & Destructor Documentation

6.754.2.1 virtual decaf::net::SocketOptions::~~SocketOptions ( ) [virtual]

### 6.754.3 Field Documentation

6.754.3.1 const int decaf::net::SocketOptions::SOCKET\_OPTION\_BINDADDR  
[static]

Fetch the local address binding of a socket (this option cannot be "set" only "gotten", since sockets are bound at creation time, and so the locally bound address cannot be changed).

The default local address of a socket is INADDR\_ANY, meaning any local address on a multi-homed host. A multi-homed host can use this option to accept connections to only one of its addresses (in the case of a **ServerSocket** (p. 3447) or DatagramSocket), or to specify its return address to the peer (for a **Socket** (p. 3607) or DatagramSocket). The parameter of this option is an **InetAddress** (p. 2077).

6.754.3.2 const int decaf::net::SocketOptions::SOCKET\_OPTION\_BROADCAST  
[static]

Sets SO\_BROADCAST for a socket.

This option enables and disables the ability of the process to send broadcast messages. It is supported for only datagram sockets and only on networks that support the concept of a broadcast message (e.g. Ethernet, token ring, etc.), and it is set by default for DatagramSockets.

**6.754.3.3** `const int decaf::net::SocketOptions::SOCKET_OPTION_IP_-MULTICAST_IF` [static]

Set which outgoing interface on which to send multicast packets.

Useful on hosts with multiple network interfaces, where applications want to use other than the system default. Takes/returns an **InetAddress** (p. 2077).

Valid for Multicast: DatagramSocketImpl.

**6.754.3.4** `const int decaf::net::SocketOptions::SOCKET_OPTION_IP_-MULTICAST_IF2` [static]

Same as above.

This option is introduced so that the behaviour with IP\_MULTICAST\_IF will be kept the same as before, while this new option can support setting outgoing interfaces with either IPv4 and IPv6 addresses.

**6.754.3.5** `const int decaf::net::SocketOptions::SOCKET_OPTION_IP_-MULTICAST_LOOP` [static]

This option enables or disables local loopback of multicast datagrams.

This option is enabled by default for Multicast Sockets.

**6.754.3.6** `const int decaf::net::SocketOptions::SOCKET_OPTION_IP_TOS` [static]

This option sets the type-of-service or traffic class field in the IP header for a TCP or UDP socket.

**6.754.3.7** `const int decaf::net::SocketOptions::SOCKET_OPTION_KEEPALIVE` [static]

When the keepalive option is set for a TCP socket and no data has been exchanged across the socket in either direction for 2 hours (NOTE: the actual value is implementation dependent), TCP automatically sends a keepalive probe to the peer.

This probe is a TCP segment to which the peer must respond. One of three responses is expected: 1. The peer responds with the expected ACK. The application is not notified (since everything is OK). TCP will send another probe following another 2 hours of inactivity. 2. The peer responds with an RST, which tells the local TCP that the peer host has crashed and rebooted. The socket is closed. 3. There is no response from the peer. The socket is closed. The purpose of this option is to detect if the peer host crashes.

Valid only for TCP socket: **SocketImpl** (p. 3635)

**6.754.3.8 const int decaf::net::SocketOptions::SOCKET\_OPTION\_LINGER**  
[static]

Specify a linger-on-close timeout.

This option disables/enables immediate return from a close() of a TCP **Socket** (p. 3607). Enabling this option with a non-zero Integer timeout means that a close() will block pending the transmission and acknowledgment of all data written to the peer, at which point the socket is closed gracefully. Upon reaching the linger timeout, the socket is closed forcefully, with a TCP RST. Enabling the option with a timeout of zero does a forceful close immediately. If the specified timeout value exceeds 65,535 it will be reduced to 65,535.

Valid only for TCP: **SocketImpl** (p. 3635)

**6.754.3.9 const int decaf::net::SocketOptions::SOCKET\_OPTION\_OOINLINE**  
[static]

When the OOINLINE option is set, any TCP urgent data received on the socket will be received through the socket input stream.

When the option is disabled (which is the default) urgent data is silently discarded.

**6.754.3.10 const int decaf::net::SocketOptions::SOCKET\_OPTION\_RCVBUF**  
[static]

Set a hint the size of the underlying buffers used by the platform for incoming network I/O.

When used in set, this is a suggestion to the kernel from the application about the size of buffers to use for the data to be received over the socket. When used in get, this must return the size of the buffer actually used by the platform when receiving in data on this socket. Valid for all sockets: **SocketImpl** (p. 3635), **DatagramSocketImpl**.

**6.754.3.11 const int decaf::net::SocketOptions::SOCKET\_OPTION\_-REUSEADDR** [static]

Sets SO\_REUSEADDR for a socket.

This is used only for MulticastSockets in decaf, and it is set by default for MulticastSockets.

**6.754.3.12 const int decaf::net::SocketOptions::SOCKET\_OPTION\_SNDBUF**  
[static]

Set a hint the size of the underlying buffers used by the platform for outgoing network I/O.

When used in set, this is a suggestion to the kernel from the application about the size of buffers to use for the data to be sent over the socket. When used in get, this must

return the size of the buffer actually used by the platform when sending out data on this socket. Valid for all sockets: **SocketImpl** (p. 3635), DatagramSocketImpl

#### 6.754.3.13 `const int decaf::net::SocketOptions::SOCKET_OPTION_TCP_NODELAY` [static]

Disable Nagle's algorithm for this connection.

Written data to the network is not buffered pending acknowledgment of previously written data. Valid for TCP sockets.

#### 6.754.3.14 `const int decaf::net::SocketOptions::SOCKET_OPTION_TIMEOUT` [static]

Set a timeout on blocking **Socket** (p. 3607) operations.

The option must be set prior to entering a blocking operation to take effect.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketOptions.h`

## 6.755 `decaf::net::SocketTimeoutException` Class Reference

```
#include <src/main/decaf/net/SocketTimeoutException.h>
```

Inheritance diagram for `decaf::net::SocketTimeoutException`:

### Public Member Functions

- **SocketTimeoutException** () throw ()  
*Default Constructor.*
- **SocketTimeoutException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **SocketTimeoutException** (const **SocketTimeoutException** &ex) throw ()  
*Copy Constructor.*
- **SocketTimeoutException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **SocketTimeoutException** (const std::exception \*cause) throw ()  
*Constructor.*

- **SocketTimeoutException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **SocketTimeoutException** \* clone () const  
*Clones this exception.*
- virtual ~**SocketTimeoutException** () throw ()

### 6.755.1 Constructor & Destructor Documentation

**6.755.1.1** decaf::net::SocketTimeoutException::SocketTimeoutException ( ) throw ()  
[inline]

Default Constructor.

**6.755.1.2** decaf::net::SocketTimeoutException::SocketTimeoutException ( const Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.755.1.3** decaf::net::SocketTimeoutException::SocketTimeoutException ( const SocketTimeoutException & ex ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.755.1.4** decaf::net::SocketTimeoutException::SocketTimeoutException ( const char \* file, const int lineNumber, const std::exception \* cause, const char \* msg, ... ) throw ()  
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
-------------	--------------------------------------

<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.755.1.5** `decaf::net::SocketTimeoutException::SocketTimeoutException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.755.1.6** `decaf::net::SocketTimeoutException::SocketTimeoutException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.755.1.7** `virtual decaf::net::SocketTimeoutException::~~SocketTimeoutException ( ) throw () [inline, virtual]`

## 6.755.2 Member Function Documentation

**6.755.2.1** `virtual SocketTimeoutException* decaf::net::SocketTimeoutException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::InterruptedIOException** (p. 2198).



The documentation for this class was generated from the following file:

- src/main/decaf/net/SocketTimeoutException.h

## 6.756 decaf::net::ssl::SSLContext Class Reference

Represents on implementation of the Secure **Socket** (p. 3607) Layer for streaming based sockets.

```
#include <src/main/decaf/net/ssl/SSLContext.h>
```

### Public Member Functions

- **SSLContext** (**SSLContextSpi** \*contextImpl)
- virtual **~SSLContext** ()
- **SocketFactory** \* **getSocketFactory** ()  
*Returns an **SocketFactory** (p. 3629) instance for use with this Context, the **SocketFactory** (p. 3629) is owned by the Context and should not be deleted by the caller.*
- **ServerSocketFactory** \* **getServerSocketFactory** ()  
*Returns an **ServerSocketFactory** (p. 3457) instance for use with this Context, the **ServerSocketFactory** (p. 3457) is owned by the Context and should not be deleted by the caller.*
- **SSLParameters** \* **getDefaultSSLParameters** ()
- **SSLParameters** \* **getSupportedSSLParameters** ()

### Static Public Member Functions

- static **SSLContext** \* **getDefault** ()  
*Gets the Default **SSLContext** (p. 3653).*
- static void **setDefault** (**SSLContext** \*context)  
*Sets the default **SSLContext** (p. 3653) to be returned from future calls to **getDefault**.*

### 6.756.1 Detailed Description

Represents on implementation of the Secure **Socket** (p. 3607) Layer for streaming based sockets. This class servers a a source of factories to be used to create new **SSL Socket** (p. 3607) instances.

Since

1.0

## 6.756.2 Constructor & Destructor Documentation

6.756.2.1 `decaf::net::ssl::SSLContext::SSLContext ( SSLContextSpi * contextImpl )`

6.756.2.2 `virtual decaf::net::ssl::SSLContext::~~SSLContext ( )` [*virtual*]

## 6.756.3 Member Function Documentation

6.756.3.1 `static SSLContext* decaf::net::ssl::SSLContext::getDefault ( )` [*static*]

Gets the Default **SSLContext** (p. 3653).

The default instance of the **SSLContext** (p. 3653) should be immediately usable without any need for the client to initialize this context.

### Returns

a pointer to the Default **SSLContext** (p. 3653) instance.

6.756.3.2 `SSLParameters* decaf::net::ssl::SSLContext::getDefaultSSLParameters ( )`

### Returns

a new instance of an **SSLParameters** (p. 3658) object containing the default set of settings for this **SSLContext** (p. 3653).

### Exceptions

<i>UnsupportedOperationException</i>	if the parameters cannot be retrieved.
--------------------------------------	--

6.756.3.3 `ServerSocketFactory* decaf::net::ssl::SSLContext::getServerSocketFactory ( )`

Returns an **ServerSocketFactory** (p. 3457) instance for use with this Context, the **ServerSocketFactory** (p. 3457) is owned by the Context and should not be deleted by the caller.

### Returns

a pointer to this **SSLContext**'s **ServerSocketFactory** (p. 3457) for creating **SSLServerSocket** (p. 3662) objects.

### Exceptions

<i>IllegalStateException</i>	if the <b>SSLContextSpi</b> (p. 3655) requires initialization but it has not yet been initialized.
------------------------------	--

**6.756.3.4 SocketFactory\* decaf::net::ssl::SSLContext::getSocketFactory ( )**

Returns an **SocketFactory** (p. 3629) instance for use with this Context, the **SocketFactory** (p. 3629) is owned by the Context and should not be deleted by the caller.

**Returns**

a pointer to this SSLContext's **SocketFactory** (p. 3629) for creating **SSLSocket** (p. 3670) objects.

**Exceptions**

<i>IllegalStateException</i>	if the <b>SSLContextSpi</b> (p. 3655) requires initialization but it has not yet been initialized.
------------------------------	--

**6.756.3.5 SSLParameters\* decaf::net::ssl::SSLContext::getSupportedSSLParameters ( )****Returns**

a new instance of an **SSLParameters** (p. 3658) object containing the complete set of settings for this **SSLContext** (p. 3653).

**Exceptions**

<i>UnsupportedOperationException</i>	if the parameters cannot be retrieved.
--------------------------------------	--

**6.756.3.6 static void decaf::net::ssl::SSLContext::setDefault ( SSLContext \* context )  
[static]**

Sets the default **SSLContext** (p. 3653) to be returned from future calls to getDefault.

The set **SSLContext** (p. 3653) must be fully initialized and usable. The caller is responsible for deleting this object before the Library shutdown methods are called.

**Exceptions**

<i>NullPointerException</i>	if the context passed is NULL.
-----------------------------	--------------------------------

The documentation for this class was generated from the following file:

- src/main/decaf/net/ssl/SSLContext.h

**6.757 decaf::net::ssl::SSLContextSpi Class Reference**

Defines the interface that should be provided by an **SSLContext** (p. 3653) provider.

```
#include <src/main/decaf/net/ssl/SSLContextSpi.h>
```

Inheritance diagram for decaf::net::ssl::SSLContextSpi:

## Public Member Functions

- virtual `~SSLContextSpi()`
- virtual void `providerInit(security::SecureRandom *random)=0`  
*Perform the initialization of this Context.*
- virtual `SSLParameters * providerGetDefaultSSLParameters()`  
*Creates a returns a new **SSLParameters** (p. 3658) instance that contains the default settings for this Providers **SSLContext** (p. 3653).*
- virtual `SSLParameters * providerGetSupportedSSLParameters()`  
*Creates and returns a new **SSLParameters** (p. 3658) instance that contains the full set of supported parameters for this **SSL Context**.*
- virtual `SocketFactory * providerGetSocketFactory()=0`  
*Returns a **SocketFactory** (p. 3629) instance that can be used to create new **SSLSocket** (p. 3670) objects.*
- virtual `ServerSocketFactory * providerGetServerSocketFactory()=0`  
*Returns a **ServerSocketFactory** (p. 3457) instance that can be used to create new **SSLServerSocket** (p. 3662) objects.*

### 6.757.1 Detailed Description

Defines the interface that should be provided by an **SSLContext** (p. 3653) provider.

Since

1.0

### 6.757.2 Constructor & Destructor Documentation

6.757.2.1 virtual decaf::net::ssl::SSLContextSpi::~SSLContextSpi( ) [virtual]

### 6.757.3 Member Function Documentation

6.757.3.1 virtual SSLParameters\* decaf::net::ssl::SSLContextSpi::providerGetDefaultSSLParameters( ) [virtual]

Creates a returns a new **SSLParameters** (p. 3658) instance that contains the default settings for this Providers **SSLContext** (p. 3653).

The returned **SSLParameters** (p. 3658) instance is required to have non-empty values in its ciphersuites and protocols.

### Returns

new **SSLParameters** (p. 3658) instance with the **SSLContext** (p. 3653) defaults.

### Exceptions

<i>UnsupportedOperationException</i>	if the defaults cannot be obtained.
--------------------------------------	-------------------------------------

**6.757.3.2** virtual **ServerSocketFactory\*** **decaf::net::ssl::SSLContextSpi::providerGetServerSocketFactory** (  
     ) [pure virtual]

Returns a **ServerSocketFactory** (p. 3457) instance that can be used to create new **SSLServerSocket** (p. 3662) objects.

The **ServerSocketFactory** (p. 3457) is owned by the Service Provider and should not be destroyed by the caller.

### Returns

**SocketFactory** (p. 3629) instance that can be used to create new **SSLServerSockets**.

### Exceptions

<i>IllegalStateException</i>	if the <b>SSLContextSpi</b> (p. 3655) object requires initialization but has not been initialized yet.
------------------------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLContextSpi** (p. 2926).

**6.757.3.3** virtual **SocketFactory\*** **decaf::net::ssl::SSLContextSpi::providerGetSocketFactory** (  
     ) [pure virtual]

Returns a **SocketFactory** (p. 3629) instance that can be used to create new **SSLSocket** (p. 3670) objects.

The **SocketFactory** (p. 3629) is owned by the Service Provider and should not be destroyed by the caller.

### Returns

**SocketFactory** (p. 3629) instance that can be used to create new **SSLSockets**.

### Exceptions

<i>IllegalStateException</i>	if the <b>SSLContextSpi</b> (p. 3655) object requires initialization but has not been initialized yet.
------------------------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLContextSpi** (p. 2926).

**6.757.3.4** `virtual SSLParameters* decaf::net::ssl::SSLContextSpi::providerGetSupportedSSLParameters ( )` [virtual]

Creates and returns a new **SSLParameters** (p. 3658) instance that contains the full set of supported parameters for this SSL Context.

The returned **SSLParameters** (p. 3658) instance is required to have non-empty values in its ciphersuites and protocols.

#### Returns

a new **SSLParameters** (p. 3658) instance with the full set of settings that are supported.

#### Exceptions

<i>UnsupportedOperationException</i>	if the supported parameters cannot be obtained.
--------------------------------------	---

**6.757.3.5** `virtual void decaf::net::ssl::SSLContextSpi::providerInit ( security::SecureRandom * random )` [pure virtual]

Perform the initialization of this Context.

#### Parameters

<i>random</i>	Pointer to an instance of a secure random number generator.
---------------	---

#### Exceptions

<i>NullPointerException</i>	if the SecureRandom instance is NULL.
<i>KeyManagementException</i>	if an error occurs while initializing the context.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLContextSpi** (p. 2927).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLContextSpi.h`

## 6.758 decaf::net::ssl::SSLParameters Class Reference

```
#include <src/main/decaf/net/ssl/SSLParameters.h>
```

## Public Member Functions

- **SSLParameters** ()

*Creates a new **SSLParameters** (p. 3658) instance with empty vectors for the protocols and the cipherSuites, the wantClientAuth and needClientAuth flags are set to false.*

- **SSLParameters** (const std::vector< std::string > &cipherSuites)

*Creates a new **SSLParameters** (p. 3658) instance with the given cipherSuites value, the protocols vector is empty and the wantClientAuth and needClientAuth flags are set to false.*

- **SSLParameters** (const std::vector< std::string > &cipherSuites, const std::vector< std::string > &protocols)

*Creates a new **SSLParameters** (p. 3658) instance with the given cipherSuites value and protocols value, the wantClientAuth and needClientAuth flags are set to false.*

- virtual ~**SSLParameters** ()

- std::vector< std::string > **getCipherSuites** () const

- void **setCipherSuites** (const std::vector< std::string > &cipherSuites)

*Sets the vector of ciphersuites.*

- std::vector< std::string > **getProtocols** () const

- void **setProtocols** (const std::vector< std::string > &protocols)

*Sets the vector of protocols.*

- bool **getWantClientAuth** () const

- void **setWantClientAuth** (bool wantClientAuth)

*Sets whether client authentication should be requested.*

- bool **getNeedClientAuth** () const

- void **setNeedClientAuth** (bool needClientAuth)

*Sets whether client authentication should be required.*

## 6.758.1 Constructor & Destructor Documentation

### 6.758.1.1 decaf::net::ssl::SSLParameters::SSLParameters ( )

Creates a new **SSLParameters** (p. 3658) instance with empty vectors for the protocols and the cipherSuites, the wantClientAuth and needClientAuth flags are set to false.

### 6.758.1.2 decaf::net::ssl::SSLParameters::SSLParameters ( const std::vector< std::string > & cipherSuites )

Creates a new **SSLParameters** (p. 3658) instance with the given cipherSuites value, the protocols vector is empty and the wantClientAuth and needClientAuth flags are set to false.

**Parameters**

<i>cipherSuites</i>	The vector of cipherSuites for this <b>SSLParameters</b> (p. 3658) instance (can be empty).
---------------------	---

**6.758.1.3** `decaf::net::ssl::SSLParameters::SSLParameters ( const std::vector< std::string > & cipherSuites, const std::vector< std::string > & protocols )`

Creates a new **SSLParameters** (p. 3658) instance with the given cipherSuites value and protocols value, the wantClientAuth and needClientAuth flags are set to false.

**Parameters**

<i>cipherSuites</i>	The vector of cipherSuites for this <b>SSLParameters</b> (p. 3658) instance (can be empty).
<i>protocols</i>	The vector of protocols for this <b>SSLParameters</b> (p. 3658) instance (can be empty).

**6.758.1.4** `virtual decaf::net::ssl::SSLParameters::~~SSLParameters ( )` [virtual]

**6.758.2 Member Function Documentation**

**6.758.2.1** `std::vector<std::string> decaf::net::ssl::SSLParameters::getCipherSuites ( ) const` [inline]

**Returns**

a copy of the vector of ciphersuites or an empty vector if none have been set.

**6.758.2.2** `bool decaf::net::ssl::SSLParameters::getNeedClientAuth ( ) const` [inline]

**Returns**

whether client authentication should be required.

**6.758.2.3** `std::vector<std::string> decaf::net::ssl::SSLParameters::getProtocols ( ) const` [inline]

**Returns**

a copy of the vector of protocols or an empty vector if none have been set.

**6.758.2.4** `bool decaf::net::ssl::SSLParameters::getWantClientAuth ( ) const` [inline]

**Returns**

whether client authentication should be requested.



**6.758.2.5** void decaf::net::ssl::SSLParameters::setCipherSuites ( const std::vector< std::string > & *cipherSuites* ) [inline]

Sets the vector of ciphersuites.

#### Parameters

<i>cipherSuites</i>	The vector of cipherSuites (can be an empty vector).
---------------------	--

**6.758.2.6** void decaf::net::ssl::SSLParameters::setNeedClientAuth ( bool *needClientAuth* ) [inline]

Sets whether client authentication should be required.

Calling this method clears the wantClientAuth flag.

#### Parameters

<i>needClientAuth</i>	whether client authentication should be required.
-----------------------	---

**6.758.2.7** void decaf::net::ssl::SSLParameters::setProtocols ( const std::vector< std::string > & *protocols* ) [inline]

Sets the vector of protocols.

#### Parameters

<i>protocols</i>	the vector of protocols (or an empty vector)
------------------	--

**6.758.2.8** void decaf::net::ssl::SSLParameters::setWantClientAuth ( bool *wantClientAuth* ) [inline]

Sets whether client authentication should be requested.

Calling this method clears the needClientAuth flag.

#### Parameters

<i>whether</i>	client authentication should be requested.
----------------	--

The documentation for this class was generated from the following file:

- src/main/decaf/net/ssl/SSLParameters.h

## 6.759 decaf::net::ssl::SSLServerSocket Class Reference

Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol.

```
#include <src/main/decaf/net/ssl/SSLServerSocket.h>
```

Inheritance diagram for decaf::net::ssl::SSLServerSocket:

### Public Member Functions

- virtual `~SSLServerSocket ()`
- virtual `std::vector< std::string > getSupportedCipherSuites () const =0`  
*Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 3662).*
- virtual `std::vector< std::string > getSupportedProtocols () const =0`  
*Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 3662) instance.*
- virtual `std::vector< std::string > getEnabledCipherSuites () const =0`  
*Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 3662).*
- virtual void `setEnabledCipherSuites (const std::vector< std::string > &suites)=0`  
*Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 3662) connection.*
- virtual `std::vector< std::string > getEnabledProtocols () const =0`  
*Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 3662).*
- virtual void `setEnabledProtocols (const std::vector< std::string > &protocols)=0`  
*Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 3662) connection.*
- virtual bool `getWantClientAuth () const =0`
- virtual void `setWantClientAuth (bool value)=0`  
*Sets whether or not this **Socket** (p. 3607) will request Client Authentication.*
- virtual bool `getNeedClientAuth () const =0`
- virtual void `setNeedClientAuth (bool value)=0`  
*Sets whether or not this **Socket** (p. 3607) will require Client Authentication.*

## Protected Member Functions

- **SSLServerSocket** ()

*Creates a non-bound server socket.*

- **SSLServerSocket** (int port)

*Creates a new **ServerSocket** (p. 3447) bound to the specified port, if the value of port is 0, then any free port is chosen.*

- **SSLServerSocket** (int port, int backlog)

*Creates a new **ServerSocket** (p. 3447) bound to the specified port, if the value of port is 0, then any free port is chosen.*

- **SSLServerSocket** (int port, int backlog, const **decaf::net::InetAddress** \*address)

*Creates a new **ServerSocket** (p. 3447) bound to the specified port, if the value of port is 0, then any free port is chosen.*

### 6.759.1 Detailed Description

Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol. The main function of this class is to create **SSLSocket** (p. 3670) objects by accepting connections from client sockets over SSL.

#### Since

1.0

### 6.759.2 Constructor & Destructor Documentation

#### 6.759.2.1 decaf::net::ssl::SSLServerSocket::SSLServerSocket ( ) [protected]

Creates a non-bound server socket.

#### 6.759.2.2 decaf::net::ssl::SSLServerSocket::SSLServerSocket ( int port ) [protected]

Creates a new **ServerSocket** (p. 3447) bound to the specified port, if the value of port is 0, then any free port is chosen.

When this constructor is called the size of the backlog queue is set at 50, connections that arrive after the backlog has been reached are refused.

If a **SocketImplFactory** (p. 3644) is registered then the `createSocketImpl` method on the factory will be called otherwise a default **SocketImpl** (p. 3635) is created.

#### Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3447) to.
-------------	--

**Exceptions**

<i>IOException</i>	if there is an I/O error while performing this operation.
<i>IllegalArgumentEx-ception</i>	if the port value is negative or greater than 65535.

### 6.759.2.3 `decaf::net::ssl::SSLServerSocket::SSLServerSocket ( int port, int backlog )` [protected]

Creates a new **ServerSocket** (p. 3447) bound to the specified port, if the value of port is 0, then any free port is chosen.

When this constructor is called the size of the backlog queue is set at backlog, connections that arrive after the backlog has been reached are refused. If backlog is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 3644) is registered then the `createSocketImpl` method on the factory will be called otherwise a default **SocketImpl** (p. 3635) is created.

**Parameters**

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3447) to.
<i>backlog</i>	The the number of incoming connection attempts to queue before connections are refused.

**Exceptions**

<i>IOException</i>	if there is an I/O error while performing this operation.
<i>IllegalArgumentEx-ception</i>	if the port value is negative or greater than 65535.

### 6.759.2.4 `decaf::net::ssl::SSLServerSocket::SSLServerSocket ( int port, int backlog, const decaf::net::InetAddress * address )` [protected]

Creates a new **ServerSocket** (p. 3447) bound to the specified port, if the value of port is 0, then any free port is chosen.

If the value of the `ifAddress` is empty or NULL then the ANY address is used.

When this constructor is called the size of the backlog queue is set at backlog, connections that arrive after the backlog has been reached are refused. If backlog is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 3644) is registered then the `createSocketImpl` method on the factory will be called otherwise a default **SocketImpl** (p. 3635) is created.

**Parameters**

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3447) to.
<i>backlog</i>	The the number of incoming connection attempts to queue before connections are refused.
<i>ifAddress</i>	The IP Address to bind to on the local machine.

**Exceptions**

<i>IOException</i>	if there is an I/O error while performing this operation.
<i>IllegalArgumentEx-ception</i>	if the port value is negative or greater than 65535.

**6.759.2.5** virtual decaf::net::ssl::SSLServerSocket::~~SSLServerSocket ( ) [virtual]

**6.759.3 Member Function Documentation**

**6.759.3.1** virtual std::vector<std::string> decaf::net::ssl::SSLServerSocket::getEnabledCipherSuites ( ) const [pure virtual]

Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 3662).

**Returns**

vector of the names of all enabled Cipher Suites.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2933).

**6.759.3.2** virtual std::vector<std::string> decaf::net::ssl::SSLServerSocket::getEnabledProtocols ( ) const [pure virtual]

Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 3662).

**Returns**

vector of the names of all enabled Protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2933).

**6.759.3.3** virtual bool decaf::net::ssl::SSLServerSocket::getNeedClientAuth ( ) const [pure virtual]

**Returns**

true if the **Socket** (p. 3607) requires client Authentication.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2933).

```
6.759.3.4 virtual std::vector<std::string> de-
caf::net::ssl::SSLServerSocket::getSupportedCipherSuites ( )
const [pure virtual]
```

Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 3662).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 3607).

#### Returns

a vector containing the names of all the supported cipher suites.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2933).

```
6.759.3.5 virtual std::vector<std::string> de-
caf::net::ssl::SSLServerSocket::getSupportedProtocols ( )
const [pure virtual]
```

Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 3662) instance.

#### Returns

a vector containing the names of all the supported protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2934).

```
6.759.3.6 virtual bool decaf::net::ssl::SSLServerSocket::getWantClientAuth ( ) const [pure
virtual]
```

#### Returns

true if the **Socket** (p. 3607) request client Authentication.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2934).

```
6.759.3.7 virtual void decaf::net::ssl::SSLServerSocket::setEnabledCipherSuites ( const
std::vector< std::string > & suites ) [pure virtual]
```

Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 3662) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

#### Parameters

<i>suites</i>	An Vector of names for all the Cipher Suites that are to be enabled.
---------------	--

**Exceptions**

<i>IllegalArgumentEx- ception</i>	if the vector is empty or one of the names is invalid.
---------------------------------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2934).

**6.759.3.8** `virtual void decaf::net::ssl::SSLServerSocket::setEnabledProtocols ( const  
std::vector< std::string > & protocols ) [pure virtual]`

Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 3662) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

**Parameters**

<i>protocols</i>	An Vector of names for all the Protocols that are to be enabled.
------------------	--

**Exceptions**

<i>IllegalArgumentEx- ception</i>	if the vector is empty or one of the names is invalid.
---------------------------------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2934).

**6.759.3.9** `virtual void decaf::net::ssl::SSLServerSocket::setNeedClientAuth ( bool value )  
[pure virtual]`

Sets whether or not this **Socket** (p. 3607) will require Client Authentication.

If set to true the **Socket** (p. 3607) (when used in server mode) will require that the client authenticate itself, if the client doesn't send authentication the socket will not allow negotiation to continue.

**Parameters**

<i>value</i>	Whether the server socket should require client authentication.
--------------	---

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2935).

**6.759.3.10** `virtual void decaf::net::ssl::SSLServerSocket::setWantClientAuth ( bool value )  
[pure virtual]`

Sets whether or not this **Socket** (p. 3607) will request Client Authentication.

If set to true the **Socket** (p. 3607) (when used in server mode) will request that the client authenticate itself, if the client doesn't send authentication the socket will still allow negotiation to continue.

**Parameters**

<i>value</i>	Whether the server socket should request client authentication.
--------------	---

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2935).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLServerSocket.h`

**6.760 decaf::net::ssl::SSLServerSocketFactory Class Reference**

Factory class interface that provides methods to create SSL Server Sockets.

```
#include <src/main/decaf/net/ssl/SSLServerSocketFactory.h>
```

Inheritance diagram for `decaf::net::ssl::SSLServerSocketFactory`:

**Public Member Functions**

- `virtual ~SSLServerSocketFactory ()`
- `virtual std::vector< std::string > getDefaultCipherSuites ()=0`  
*Returns the list of cipher suites which are enabled by default.*
- `virtual std::vector< std::string > getSupportedCipherSuites ()=0`  
*Returns the names of the cipher suites which could be enabled for use on an SSL connection.*

**Static Public Member Functions**

- `static ServerSocketFactory * getDefault ()`  
*Returns the current default SSL **ServerSocketFactory** (p. 3457), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.*

**Protected Member Functions**

- `SSLServerSocketFactory ()`

**6.760.1 Detailed Description**

Factory class interface that provides methods to create SSL Server Sockets.

**Since**

1.0



## 6.760.2 Constructor & Destructor Documentation

6.760.2.1 `decaf::net::ssl::SSLServerSocketFactory::SSLServerSocketFactory ( )`  
[protected]

6.760.2.2 `virtual decaf::net::ssl::SSLServerSocketFactory::~~SSLServerSocketFactory ( )`  
[virtual]

## 6.760.3 Member Function Documentation

6.760.3.1 `static ServerSocketFactory* decaf::net::ssl::SSLServerSocketFactory::getDefault ( )` [static]

Returns the current default SSL **ServerSocketFactory** (p. 3457), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.

This method returns **SSLContext::getDefault()** (p. 3654)->**getServerSocketFactory()**. If that call fails, a non-functional factory is returned.

### Returns

the default SSL **ServerSocketFactory** (p. 3457) pointer.

### See also

**decaf::net::ssl::SSLContext::getDefault()** (p. 3654)

Reimplemented from **decaf::net::ServerSocketFactory** (p. 3460).

6.760.3.2 `virtual std::vector<std::string> decaf::net::ssl::SSLServerSocketFactory::getDefaultCipherSuites ( )` [pure virtual]

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

### Returns

an STL vector containing the list of cipher suites enabled by default.

### See also

**getSupportedCipherSuites()** (p. 3670)

Implemented in **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1749), and **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2940).

**6.760.3.3** `virtual std::vector<std::string> decaf::net::ssl::SSLServerSocketFactory::getSupportedCipherSuites ( ) [pure virtual]`

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

### Returns

an STL vector containing the list of supported cipher suites.

### See also

`getDefaultCipherSuites()` (p. 3669)

Implemented in `decaf::internal::net::ssl::DefaultSSLServerSocketFactory` (p. 1749), and `decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory` (p. 2940).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLServerSocketFactory.h`

## 6.761 decaf::net::ssl::SSLSocket Class Reference

```
#include <src/main/decaf/net/ssl/SSLSocket.h>
```

Inheritance diagram for `decaf::net::ssl::SSLSocket`:

### Public Member Functions

- **SSLSocket** ()
- **SSLSocket** (const **InetAddress** \*address, int port)  
*Creates a new **SSLSocket** (p. 3670) instance and connects it to the given address and port.*
- **SSLSocket** (const **InetAddress** \*address, int port, const **InetAddress** \*localAddress, int localPort)  
*Creates a new **SSLSocket** (p. 3670) instance and connects it to the given address and port.*
- **SSLSocket** (const std::string &host, int port)  
*Creates a new **SSLSocket** (p. 3670) instance and connects it to the given host and port.*

- **SSLSocket** (const std::string &host, int port, const **InetAddress** \*localAddress, int localPort)  
*Creates a new **SSLSocket** (p. 3670) instance and connects it to the given host and port.*
- virtual ~**SSLSocket** ()
- virtual std::vector< std::string > **getSupportedCipherSuites** () const =0  
*Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 3670).*
- virtual std::vector< std::string > **getSupportedProtocols** () const =0  
*Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 3670) instance.*
- virtual std::vector< std::string > **getEnabledCipherSuites** () const =0  
*Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSL Socket** (p. 3607).*
- virtual void **setEnabledCipherSuites** (const std::vector< std::string > &suites)=0  
*Sets the Cipher Suites that are to be enabled on the **SSL Socket** (p. 3607) connection.*
- virtual std::vector< std::string > **getEnabledProtocols** () const =0  
*Returns a vector containing the names of all the currently enabled Protocols for this **SSL Socket** (p. 3607).*
- virtual void **setEnabledProtocols** (const std::vector< std::string > &protocols)=0  
*Sets the Protocols that are to be enabled on the **SSL Socket** (p. 3607) connection.*
- virtual **SSLParameters** **getSSLParameters** () const  
*Returns an **SSLParameters** (p. 3658) object for this **SSLSocket** (p. 3670) instance.*
- virtual void **setSSLParameters** (const **SSLParameters** &value)  
*Sets the **SSLParameters** (p. 3658) for this **SSLSocket** (p. 3670) using the supplied **SSLParameters** (p. 3658) instance.*
- virtual void **startHandshake** ()=0  
*Initiates a handshake for this **SSL Connection**, this can be necessary for several reasons such as using new encryption keys, or starting a new session.*
- virtual void **setUseClientMode** (bool value)=0  
*Determines the mode that the socket uses when a handshake is initiated, client or server.*
- virtual bool **getUseClientMode** () const =0

*Gets whether this **Socket** (p. 3607) is in Client or Server mode, true indicates that the mode is set to Client.*

- virtual void **setNeedClientAuth** (bool value)=0  
*Sets the **Socket** (p. 3607) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*
- virtual bool **getNeedClientAuth** () const =0  
*Returns if this socket is configured to require client authentication, true means that it has and that clients that failed to authenticate will be rejected.*
- virtual void **setWantClientAuth** (bool value)=0  
*Sets the **Socket** (p. 3607) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*
- virtual bool **getWantClientAuth** () const =0  
*Returns if this socket is configured to request client authentication, true means that it has and that clients that failed to authenticate will be rejected but that clients that do not send a certificate are not considered to have failed authentication.*

### 6.761.1 Detailed Description

Since

1.0

### 6.761.2 Constructor & Destructor Documentation

6.761.2.1 **decaf::net::ssl::SSLSocket::SSLSocket ( )**

6.761.2.2 **decaf::net::ssl::SSLSocket::SSLSocket ( const InetAddress \* address, int port )**

Creates a new **SSLSocket** (p. 3670) instance and connects it to the given address and port.

If the host parameter is empty then the loop back address is used.

#### Parameters

<i>address</i>	The address to connect to.
<i>port</i>	The port number to connect to [0...65535]

#### Exceptions

<b>UnknownHostException</b> (p. 4019)	if the host cannot be resolved.
<b>IOException</b>	if an I/O error occurs while connecting the <b>Socket</b> (p. 3607).

<i>NullPointerException</i>	if the <b>InetAddress</b> (p. 2077) instance is NULL.
<i>IllegalArgumentException</i>	if the port is not in range [0...65535]

#### 6.761.2.3 decaf::net::ssl::SSLSocket::SSLSocket ( const InetAddress \* address, int port, const InetAddress \* localAddress, int localPort )

Creates a new **SSLSocket** (p. 3670) instance and connects it to the given address and port.

The **Socket** (p. 3607) will also bind to the local address and port specified.

##### Parameters

<i>address</i>	The address to connect to.
<i>port</i>	The port number to connect to [0...65535]
<i>localAddress</i>	The IP address on the local machine to bind to.
<i>localPort</i>	The port on the local machine to bind to.

##### Exceptions

<i>UnknownHostException</i> (p. 4019)	if the host cannot be resolved.
<i>IOException</i>	if an I/O error occurs while connecting the <b>Socket</b> (p. 3607).
<i>NullPointerException</i>	if the <b>InetAddress</b> (p. 2077) instance is NULL.
<i>IllegalArgumentException</i>	if the port is not in range [0...65535]

#### 6.761.2.4 decaf::net::ssl::SSLSocket::SSLSocket ( const std::string & host, int port )

Creates a new **SSLSocket** (p. 3670) instance and connects it to the given host and port.

If the host parameter is empty then the loop back address is used.

##### Parameters

<i>host</i>	The host name or IP address to connect to, empty string means loopback.
<i>port</i>	The port number to connect to [0...65535]

##### Exceptions

<i>UnknownHostException</i> (p. 4019)	if the host cannot be resolved.
--	---------------------------------

<i>IOException</i>	if an I/O error occurs while connecting the <b>Socket</b> (p. 3607).
<i>IllegalArgumentEx- ception</i>	if the port if not in range [0...65535]

#### 6.761.2.5 `decaf::net::ssl::SSLSocket::SSLSocket ( const std::string & host, int port, const InetAddress * localAddress, int localPort )`

Creates a new **SSLSocket** (p. 3670) instance and connects it to the given host and port.

If the host parameter is empty then the loop back address is used.

##### Parameters

<i>host</i>	The host name or IP address to connect to, empty string means loopback.
<i>port</i>	The port number to connect to [0...65535]
<i>localAd- dress</i>	The IP address on the local machine to bind to.
<i>localPort</i>	The port on the local machine to bind to.

##### Exceptions

<i>UnknownHostEx- ception</i> (p. 4019)	if the host cannot be resolved.
<i>IOException</i>	if an I/O error occurs while connecting the <b>Socket</b> (p. 3607).
<i>IllegalArgumentEx- ception</i>	if the port if not in range [0...65535]

#### 6.761.2.6 `virtual decaf::net::ssl::SSLSocket::~~SSLSocket ( ) [virtual]`

### 6.761.3 Member Function Documentation

#### 6.761.3.1 `virtual std::vector<std::string> decaf::net::ssl::SSLSocket::getEnabledCipherSuites ( ) const [pure virtual]`

Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSL Socket** (p. 3607).

##### Returns

vector of the names of all enabled Cipher Suites.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2947).

**6.761.3.2** `virtual std::vector<std::string> decaf::net::ssl::SSLSocket::getEnabledProtocols ( ) const [pure virtual]`

Returns a vector containing the names of all the currently enabled Protocols for this **SSL Socket** (p. 3607).

#### Returns

vector of the names of all enabled Protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2948).

**6.761.3.3** `virtual bool decaf::net::ssl::SSLSocket::getNeedClientAuth ( ) const [pure virtual]`

Returns if this socket is configured to require client authentication, true means that it has and that clients that failed to authenticate will be rejected.

This option is only useful when the socket is operating in server mode.

#### Returns

true if client authentication is required.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2948).

**6.761.3.4** `virtual SSLParameters decaf::net::ssl::SSLSocket::getSSLParameters ( ) const [virtual]`

Returns an **SSLParameters** (p. 3658) object for this **SSLSocket** (p. 3670) instance.

The cipherSuites and protocols vectors in the returned **SSLParameters** (p. 3658) reference will never be empty.

#### Returns

an **SSLParameters** (p. 3658) object with the settings in use for the **SSLSocket** (p. 3670).

**6.761.3.5** `virtual std::vector<std::string> decaf::net::ssl::SSLSocket::getSupportedCipherSuites ( ) const [pure virtual]`

Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 3670).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 3607).

#### Returns

a vector containing the names of all the supported cipher suites.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2949).

**6.761.3.6** `virtual std::vector<std::string> decaf::net::ssl::SSLSocket::getSupportedProtocols ( ) const [pure virtual]`

Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 3670) instance.

#### Returns

a vector containing the names of all the supported protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2949).

**6.761.3.7** `virtual bool decaf::net::ssl::SSLSocket::getUseClientMode ( ) const [pure virtual]`

Gets whether this **Socket** (p. 3607) is in Client or Server mode, true indicates that the mode is set to Client.

#### Returns

true if the **Socket** (p. 3607) is in Client mode, false otherwise.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2950).

**6.761.3.8** `virtual bool decaf::net::ssl::SSLSocket::getWantClientAuth ( ) const [pure virtual]`

Returns if this socket is configured to request client authentication, true means that it has and that clients that failed to authenticate will be rejected but that clients that do not send a certificate are not considered to have failed authentication.

This option is only useful when the socket is operating in server mode.

#### Returns

true if client authentication is required.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2950).

**6.761.3.9** `virtual void decaf::net::ssl::SSLSocket::setEnabledCipherSuites ( const std::vector<std::string> & suites ) [pure virtual]`

Sets the Cipher Suites that are to be enabled on the **SSL Socket** (p. 3607) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.



**Parameters**

<i>suites</i>	An Vector of names for all the Cipher Suites that are to be enabled.
---------------	--

**Exceptions**

<i>IllegalArgumentEx- ception</i>	if the vector is empty or one of the names is invalid.
---------------------------------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2951).

**6.761.3.10** `virtual void decaf::net::ssl::SSLSocket::setEnabledProtocols ( const std::vector< std::string > & protocols ) [pure virtual]`

Sets the Protocols that are to be enabled on the SSL **Socket** (p. 3607) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

**Parameters**

<i>protocols</i>	An Vector of names for all the Protocols that are to be enabled.
------------------	--

**Exceptions**

<i>IllegalArgumentEx- ception</i>	if the vector is empty or one of the names is invalid.
---------------------------------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2951).

**6.761.3.11** `virtual void decaf::net::ssl::SSLSocket::setNeedClientAuth ( bool value ) [pure virtual]`

Sets the **Socket** (p. 3607) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled an the client does not provide a certificate then the handshake is considered failed and the connection is refused. Calling this method resets any previous value for this option as well as clears any value set in the setWantClientAuth method.

**Parameters**

<i>value</i>	The value indicating if a client is required to authenticate itself or not.
--------------	---

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2952).

**6.761.3.12** `virtual void decaf::net::ssl::SSLSocket::setSSLParameters ( const SSLParameters & value )` [virtual]

Sets the **SSLParameters** (p. 3658) for this **SSLSocket** (p. 3670) using the supplied **SSLParameters** (p. 3658) instance.

If the cipherSuites vector in the **SSLParameters** (p. 3658) instance is not empty then the setEnabledCipherSuites method is called with that vector, if the protocols vector in the **SSLParameters** (p. 3658) instance is not empty then the setEnabledProtocols method is called with that vector. If the needClientAuth value or the wantClientAuth value is true then the setNeedClientAuth and setWantClientAuth methods are called respectively with a value of true, otherwise the setWantClientAuth method is called with a value of false.

#### Parameters

<i>value</i>	The <b>SSLParameters</b> (p. 3658) instance that is used to update this <b>SSLSocket</b> 's settings.
--------------	---

#### Exceptions

<i>IllegalArgumentEx- ception</i>	if an error occurs while calling setEnabledCipherSuites or setEnabledProtocols.
---------------------------------------	---

**6.761.3.13** `virtual void decaf::net::ssl::SSLSocket::setUseClientMode ( bool value )` [pure virtual]

Determines the mode that the socket uses when a handshake is initiated, client or server.

This method must be called prior to any handshake attempts on this **Socket** (p. 3607), once a handshake has been initiated this socket remains in the set mode; client or server, for the life of this object.

#### Parameters

<i>value</i>	The mode setting, true for client or false for server.
--------------	--

#### Exceptions

<i>IllegalArgumentEx- ception</i>	if the handshake process has begun and mode is locked.
---------------------------------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2953).

**6.761.3.14** `virtual void decaf::net::ssl::SSLSocket::setWantClientAuth ( bool value )` [pure virtual]

Sets the **Socket** (p. 3607) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled and the client does not provide a certificate then the handshake is considered to have succeeded, if it does send a certificate and that certificate is invalid the the handshake will fail. Calling this method resets any previous value for this option as well as clears any value set in the `setNeedClientAuth` method.

#### Parameters

<i>value</i>	The value indicating if a client is requested to authenticate itself or not.
--------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2953).

**6.761.3.15** `virtual void decaf::net::ssl::SSLSocket::startHandshake ( )` [pure virtual]

Initiates a handshake for this SSL Connection, this can be necessary for several reasons such as using new encryption keys, or starting a new session.

When called for the first time after the socket connects this method blocks until the handshake is completed. The provider is not required to support multiple handshakes and can throw an `IOException` to indicate an error.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while performing the Handshake
--------------------	---

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2954).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLSocket.h`

## 6.762 decaf::net::ssl::SSLSocketFactory Class Reference

Factory class interface for a **SocketFactory** (p. 3629) that can create **SSLSocket** (p. 3670) objects.

```
#include <src/main/decaf/net/ssl/SSLSocketFactory.h>
```

Inheritance diagram for `decaf::net::ssl::SSLSocketFactory`:

#### Public Member Functions

- `virtual ~SSLSocketFactory ()`
- `virtual std::vector< std::string > getDefaultCipherSuites ()=0`  
Returns the list of cipher suites which are enabled by default.

- virtual `std::vector< std::string > getSupportedCipherSuites ()=0`  
*Returns the names of the cipher suites which could be enabled for use on an SSL connection.*
- virtual `Socket * createSocket (Socket *socket, std::string host, int port, bool autoClose)=0`  
*Returns a socket layered over an existing socket connected to the named host, at the given port.*

### Static Public Member Functions

- static `SocketFactory * getDefault ()`  
*Returns the current default SSL **SocketFactory** (p. 3629), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.*

### Protected Member Functions

- `SSLSocketFactory ()`

#### 6.762.1 Detailed Description

Factory class interface for a **SocketFactory** (p. 3629) that can create **SSLSocket** (p. 3670) objects.

Since

1.0

#### 6.762.2 Constructor & Destructor Documentation

6.762.2.1 `decaf::net::ssl::SSLSocketFactory::SSLSocketFactory ( )` [protected]

6.762.2.2 `virtual decaf::net::ssl::SSLSocketFactory::~SSLSocketFactory ( )` [virtual]

#### 6.762.3 Member Function Documentation

6.762.3.1 `virtual Socket* decaf::net::ssl::SSLSocketFactory::createSocket ( Socket * socket, std::string host, int port, bool autoClose )` [pure virtual]

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

**Parameters**

<i>socket</i>	The existing socket to layer over.
<i>host</i>	The server host the original <b>Socket</b> (p. 3607) is connected to.
<i>port</i>	The server port the original <b>Socket</b> (p. 3607) is connected to.
<i>autoClose</i>	Should the layered over <b>Socket</b> (p. 3607) be closed when the topmost socket is closed.

**Returns**

a new **Socket** (p. 3607) instance that wraps the given **Socket** (p. 3607).

**Exceptions**

<i>IOException</i>	if an I/O exception occurs while performing this operation.
<i>UnknownHostException</i> (p. 4019)	if the host is unknown.

Implemented in **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1753), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2962).

**6.762.3.2 static SocketFactory\* decaf::net::ssl::SSLSocketFactory::getDefault ( )**  
[static]

Returns the current default SSL **SocketFactory** (p. 3629), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.

This method returns **SSLContext::getDefault()** (p. 3654)->**getSocketFactory()**. If that call fails, a non-functional factory is returned.

**Returns**

the default SSL **SocketFactory** (p. 3629) pointer.

**See also**

**decaf::net::ssl::SSLContext::getDefault()** (p. 3654)

Reimplemented from **decaf::net::SocketFactory** (p. 3633).

**6.762.3.3 virtual std::vector<std::string> decaf::net::ssl::SSLSocketFactory::getDefaultCipherSuites ( )**  
[pure virtual]

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

**Returns**

an STL vector containing the list of cipher suites enabled by default.

**See also**

**getSupportedCipherSuites()** (p. 3682)

Implemented in **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1756), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2965).

```
6.762.3.4  virtual std::vector<std::string> de-
            caf::net::ssl::SSLSocketFactory::getSupportedCipherSuites ( )
            [pure virtual]
```

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

**Returns**

an STL vector containing the list of supported cipher suites.

**See also**

**getDefaultCipherSuites()** (p. 3681)

Implemented in **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1757), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2966).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLSocketFactory.h`

## 6.763 activemq::transport::tcp::SslTransport Class Reference

**Transport** (p. 3996) for connecting to a Broker using an SSL Socket.

```
#include <src/main/activemq/transport/tcp/SslTransport.h>
```

Inheritance diagram for `activemq::transport::tcp::SslTransport`:

**Public Member Functions**

- **SslTransport** (const **Pointer**< **Transport** > &next)

*Creates a new instance of the **SslTransport** (p. 3682), the transport will not attempt to connect to a remote host until the connect method is called.*

- virtual `~SslTransport()`

### Protected Member Functions

- virtual `decaf::net::Socket * createSocket()`

*Create an unconnected Socket instance to be used by the transport to communicate with the broker.*

#### Returns

*a newly created unconnected Socket instance.*

#### Exceptions

IOException	<i>if there is an error while creating the unconnected Socket.</i>
-------------	--

- virtual void `configureSocket(decaf::net::Socket *socket, decaf::util::Properties &properties)`

### 6.763.1 Detailed Description

**Transport** (p. 3996) for connecting to a Broker using an SSL Socket. This transport simply wraps the **TcpTransport** (p. 3865) and provides the **TcpTransport** (p. 3865) an SSL based Socket pointer allowing the core **TcpTransport** (p. 3865) logic to be reused.

#### Since

3.2.0

### 6.763.2 Constructor & Destructor Documentation

#### 6.763.2.1 `activemq::transport::tcp::SslTransport::SslTransport(const Pointer<Transport> &next)`

Creates a new instance of the **SslTransport** (p. 3682), the transport will not attempt to connect to a remote host until the connect method is called.

#### Parameters

<i>next</i>	the next transport in the chain
-------------	---------------------------------

6.763.2.2 `virtual activemq::transport::tcp::SslTransport::~~SslTransport ( ) [virtual]`

### 6.763.3 Member Function Documentation

6.763.3.1 `virtual void activemq::transport::tcp::SslTransport::configureSocket ( decaf::net::Socket * socket, decaf::util::Properties & properties ) [protected, virtual]`

6.763.3.2 `virtual decaf::net::Socket* activemq::transport::tcp::SslTransport::createSocket ( ) [protected, virtual]`

Create an unconnected Socket instance to be used by the transport to communicate with the broker.

#### Returns

a newly created unconnected Socket instance.

#### Exceptions

<i>IOException</i>	if there is an error while creating the unconnected Socket.
--------------------	---

Reimplemented from `activemq::transport::tcp::TcpTransport` (p. 3868).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/tcp/SslTransport.h`

## 6.764 `activemq::transport::tcp::SslTransportFactory` Class Reference

```
#include <src/main/activemq/transport/tcp/SslTransportFactory.h>
```

Inheritance diagram for `activemq::transport::tcp::SslTransportFactory`:

### Public Member Functions

- `virtual ~SslTransportFactory ()`

### Protected Member Functions

- `virtual Pointer< Transport > doCreateComposite (const decaf::net::URI &location, const Pointer< wireformat::WireFormat > &wireFormat, const decaf::util::Properties &properties) throw ( exceptions::ActiveMQException )`

*Creates a slimmed down **Transport** (p. 3996) instance which can be used in composite transport instances.*



## 6.765 activemq::commands::BrokerError::StackTraceElement Struct Reference 3689

### 6.764.1 Constructor & Destructor Documentation

6.764.1.1 virtual activemq::transport::tcp::SslTransportFactory::~SslTransportFactory ( )  
[virtual]

### 6.764.2 Member Function Documentation

6.764.2.1 virtual Pointer<Transport> activemq::transport::tcp::SslTransportFactory::doCreateComposite ( const decaf::net::URI & *location*, const Pointer<wireformat::WireFormat> & *wireFormat*, const decaf::util::Properties & *properties* ) throw ( exceptions::ActiveMQException ) [protected, virtual]

Creates a slimed down **Transport** (p. 3996) instance which can be used in composite transport instances.

#### Parameters

<i>location</i>	- URI location to connect to.
<i>wireFormat</i>	- the assigned WireFormat for the new <b>Transport</b> (p. 3996).
<i>properties</i>	- Properties to apply to the transport.

#### Returns

new Pointer to a **SslTransport** (p. 3682).

#### Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Reimplemented from **activemq::transport::tcp::TcpTransportFactory** (p. 3871).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/tcp/SslTransportFactory.h

## 6.765 activemq::commands::BrokerError::StackTraceElement Struct Reference

```
#include <src/main/activemq/commands/BrokerError.h>
```

#### Data Fields

- std::string **ClassName**
- std::string **FileName**
- std::string **MethodName**
- int **LineNumber**

### 6.765.1 Field Documentation

6.765.1.1 `std::string activemq::commands::BrokerError::StackTraceElement::ClassName`

6.765.1.2 `std::string activemq::commands::BrokerError::StackTraceElement::FileName`

6.765.1.3 `int activemq::commands::BrokerError::StackTraceElement::LineNumber`

6.765.1.4 `std::string activemq::commands::BrokerError::StackTraceElement::MethodName`

The documentation for this struct was generated from the following file:

- `src/main/activemq/commands/BrokerError.h`

### 6.766 `decaf::internal::io::StandardErrorOutputStream` Class Reference

Wrapper Around the Standard error Output facility on the current platform.

```
#include <src/main/decaf/internal/io/StandardErrorOutputStream.h>
```

Inheritance diagram for `decaf::internal::io::StandardErrorOutputStream`:

#### Public Member Functions

- `StandardErrorOutputStream ()`
- `virtual ~StandardErrorOutputStream ()`
- `virtual void flush () throw ( decaf::io::IOException )`

*Flushes this stream by writing any buffered output to the underlying stream.*

**Exceptions**

<b>IOException</b> (p. 2210)   <i>if an I/O error occurs.</i>
---

*The default implementation of this method does nothing.*

- `virtual void close () throw ( decaf::io::IOException )`

*Closes this object and deallocates the appropriate resources.*

*The object is generally no longer usable after calling close.*

**Exceptions**

<b>IOException</b> (p. 2210)   <i>if an error occurs while closing.</i>
---

*The default implementation of this method does nothing.*

## Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### 6.766.1 Detailed Description

Wrapper Around the Standard error Output facility on the current platform. This allows for the use of alternate output methods on platforms or compilers that do not support `std::cerr`.

### 6.766.2 Constructor & Destructor Documentation

6.766.2.1 `decaf::internal::io::StandardErrorOutputStream::StandardErrorOutputStream ( )`

6.766.2.2 `virtual decaf::internal::io::StandardErrorOutputStream::~~StandardErrorOutputStream ( ) [virtual]`

### 6.766.3 Member Function Documentation

6.766.3.1 `virtual void decaf::internal::io::StandardErrorOutputStream::close ( ) throw ( decaf::io::IOException ) [virtual]`

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

## Exceptions

<i><b>IOException</b></i> (p. 2210)	if an error occurs while closing.
--	-----------------------------------

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2993).

6.766.3.2 `virtual void decaf::internal::io::StandardErrorOutputStream::doWriteArrayBounded ( const unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [protected, virtual]`

Reimplemented from **decaf::io::OutputStream** (p. 2994).

6.766.3.3 `virtual void decaf::internal::io::StandardOutputStream::doWriteByte ( unsigned char value ) throw ( decaf::io::IOException )` [protected, virtual]

Implements `decaf::io::OutputStream` (p. 2994).

6.766.3.4 `virtual void decaf::internal::io::StandardOutputStream::flush ( ) throw ( decaf::io::IOException )` [virtual]

Flushes this stream by writing any buffered output to the underlying stream.

#### Exceptions

<i><b>IOException</b></i> (p. 2210)	if an I/O error occurs.
--	-------------------------

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from `decaf::io::OutputStream` (p. 2994).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/io/StandardOutputStream.h`

## 6.767 decaf::internal::io::StandardInputStream Class Reference

```
#include <src/main/decaf/internal/io/StandardInputStream.h>
```

Inheritance diagram for `decaf::internal::io::StandardInputStream`:

#### Public Member Functions

- `StandardInputStream ()`
- `virtual ~StandardInputStream ()`
- `virtual int available () const throw ( decaf::io::IOException )`

*Indicates the number of bytes available.*

#### Protected Member Functions

- `virtual int doReadByte () throw ( decaf::io::IOException )`

### 6.767.1 Constructor & Destructor Documentation

6.767.1.1 `decaf::internal::io::StandardInputStream::StandardInputStream ( )`

6.767.1.2 `virtual decaf::internal::io::StandardInputStream::~~StandardInputStream ( )`  
[virtual]

### 6.767.2 Member Function Documentation

6.767.2.1 `virtual int decaf::internal::io::StandardInputStream::available ( ) const throw ( decaf::io::IOException )` [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

#### Returns

the number of bytes available on this input stream.

#### Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

Reimplemented from `decaf::io::InputStream` (p. 2107).

6.767.2.2 `virtual int decaf::internal::io::StandardInputStream::doReadByte ( ) throw ( decaf::io::IOException )` [protected, virtual]

Implements `decaf::io::InputStream` (p. 2109).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/io/StandardInputStream.h`

## 6.768 decaf::internal::io::StandardOutputStream Class Reference

```
#include <src/main/decaf/internal/io/StandardOutputStream.h>
```

Inheritance diagram for `decaf::internal::io::StandardOutputStream`:

## Public Member Functions

- **StandardOutputStream** ()
- virtual **~StandardOutputStream** ()
- virtual void **flush** () throw ( decaf::io::IOException )

*Flushes this stream by writing any buffered output to the underlying stream.*

### Exceptions

<b>IOException</b> (p. 2210)	if an I/O error occurs.
------------------------------	-------------------------

*The default implementation of this method does nothing.*

- virtual void **close** () throw ( decaf::io::IOException )

*Closes this object and deallocates the appropriate resources.*

*The object is generally no longer usable after calling close.*

### Exceptions

<b>IOException</b> (p. 2210)	if an error occurs while closing.
------------------------------	-----------------------------------

*The default implementation of this method does nothing.*

## Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

## 6.768.1 Constructor & Destructor Documentation

6.768.1.1 **decaf::internal::io::StandardOutputStream::StandardOutputStream** ( )

6.768.1.2 **virtual decaf::internal::io::StandardOutputStream::~~StandardOutputStream** ( )  
[virtual]

## 6.768.2 Member Function Documentation

6.768.2.1 **virtual void decaf::internal::io::StandardOutputStream::close** ( ) throw ( **decaf::io::IOException** ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

### Exceptions

<b>IOException</b> (p. 2210)	if an error occurs while closing.
---------------------------------	-----------------------------------

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2993).

**6.768.2.2** virtual void decaf::internal::io::StandardOutputStream::doWriteArrayBounded ( const unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
[protected, virtual]

Reimplemented from **decaf::io::OutputStream** (p. 2994).

**6.768.2.3** virtual void decaf::internal::io::StandardOutputStream::doWriteByte ( unsigned char *value* ) throw ( decaf::io::IOException ) [protected, virtual]

Implements **decaf::io::OutputStream** (p. 2994).

**6.768.2.4** virtual void decaf::internal::io::StandardOutputStream::flush ( ) throw ( decaf::io::IOException ) [virtual]

Flushes this stream by writing any buffered output to the underlying stream.

### Exceptions

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
--	-------------------------

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2994).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/io/StandardOutputStream.h

## 6.769 cms::Startable Class Reference

Interface for a class that implements the start method.

```
#include <src/main/cms/Startable.h>
```

Inheritance diagram for cms::Startable:

## Public Member Functions

- virtual `~Startable()`
- virtual void `start()`=0 throw ( `CMSEException` )

*Starts the service.*

### 6.769.1 Detailed Description

Interface for a class that implements the start method. An object that implements the **Startable** (p. 3691) interface implies that until its start method is called it will be considered to be in a closed or stopped state and will throw an Exception to indicate that it is not in an started state if one of its methods is called.

Since

1.0

### 6.769.2 Constructor & Destructor Documentation

6.769.2.1 virtual `cms::Startable::~~Startable()` [inline, virtual]

### 6.769.3 Member Function Documentation

6.769.3.1 virtual void `cms::Startable::start()` throw ( `CMSEException` ) [pure virtual]

Starts the service.

### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	if an internal error occurs while starting.
--	---

Implemented in **activemq::core::ActiveMQConnection** (p. 280).

The documentation for this class was generated from the following file:

- `src/main/cms/Startable.h`

## 6.770 decaf::lang::STATIC\_CAST\_TOKEN Struct Reference

```
#include <src/main/decaf/lang/Pointer.h>
```

The documentation for this struct was generated from the following file:

- `src/main/decaf/lang/Pointer.h`



## 6.771 activemq::core::ActiveMQConstants::StaticInitializer Class Reference

```
#include <src/main/activemq/core/ActiveMQConstants.h>
```

### Public Member Functions

- **StaticInitializer** ()
- virtual **~StaticInitializer** ()

### Static Public Attributes

- static std::string **destOptions** [NUM\_OPTIONS]
- static std::string **uriParams** [NUM\_PARAMS]
- static std::map< std::string, **DestinationOption** > **destOptionMap**
- static std::map< std::string, **URIParam** > **uriParamsMap**

### 6.771.1 Constructor & Destructor Documentation

6.771.1.1 **activemq::core::ActiveMQConstants::StaticInitializer::StaticInitializer** ( )

6.771.1.2 **virtual activemq::core::ActiveMQConstants::StaticInitializer::~~StaticInitializer** ( )  
[inline, virtual]

### 6.771.2 Field Documentation

6.771.2.1 **std::map<std::string, DestinationOption>**  
**activemq::core::ActiveMQConstants::StaticInitializer::destOptionMap**  
[static]

6.771.2.2 **std::string activemq::core::ActiveMQConstants::StaticInitializer::destOptions**[NUM\_OPTIONS]  
[static]

6.771.2.3 **std::string activemq::core::ActiveMQConstants::StaticInitializer::uriParams**[NUM\_PARAMS]  
[static]

6.771.2.4 **std::map<std::string, URIParam>** **activemq::core::ActiveMQConstants::StaticInitializer::uriParamsMap**  
[static]

The documentation for this class was generated from the following file:

- **src/main/activemq/core/ActiveMQConstants.h**

## 6.772 decaf::util::StlList< E > Class Template Reference

**List** (p. 2409) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.

```
#include <src/main/decaf/util/StlList.h>
```

Inheritance diagram for decaf::util::StlList< E >:

### Data Structures

- class **ConstStlListIterator**
- class **StlListIterator**

### Public Member Functions

- **StlList** ()  
*Default constructor - does nothing.*
- **StlList** (const **StlList** &source)  
*Copy constructor - copies the content of the given set into this one.*
- **StlList** (const **Collection**< E > &source)  
*Copy constructor - copies the content of the given set into this one.*
- virtual **~StlList** ()
- virtual bool **equals** (const **StlList** &source) const
- virtual **Iterator**< E > \* **iterator** ()  
*Returns*  
*an iterator over a set of elements of type T.*
- virtual **Iterator**< E > \* **iterator** () const
- virtual **ListIterator**< E > \* **listIterator** ()  
*Returns*  
*a list iterator over the elements in this list (in proper sequence).*
- virtual **ListIterator**< E > \* **listIterator** () const
- virtual **ListIterator**< E > \* **listIterator** (std::size\_t index) throw ( decaf::lang::exceptions::IndexOutOfBo  
)

#### Parameters

index	<i>index of first element to be returned from the list iterator (by a call to the next method).</i>
-------	---

**Returns**

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

**Exceptions**

IndexOutOfBoundsException	if the index is out of range ( $\text{index} < 0 \parallel \text{index} > \text{size}()$ (p. 1226))
---------------------------	---

- virtual **ListIterator**< E > \* **listIterator** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual void **copy** (const **StlList** &source)

- virtual void **clear** () throw ( lang::exceptions::UnsupportedOperationException )

*Removes all of the elements from this collection (optional operation).*

*The collection will be empty after this method returns.*

*This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 2223) operation. Most implementations will probably choose to override this method for efficiency.*

*Note that this implementation will throw an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.*

**Exceptions**

UnsupportedOperationException	if the clear operation is not supported by this collection
-------------------------------	--

- virtual bool **contains** (const E &value) const throw ( lang::Exception )

*Returns true if this collection contains the specified element.*

*This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.*

**Parameters**

value	- the value whose presence is to be queried for in this <b>Collection</b> (p. 1216).
-------	--

**Returns**

*true if the value is contained in this collection*

**Exceptions**

Exception	if an error occurs,
-----------	---------------------

- virtual std::size\_t **indexOf** (const E &value) throw ( decaf::lang::exceptions::NoSuchElementException )

*Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.*

*More formally, returns the lowest index i such that  $\text{get}(i) == \text{value}$ , or -1 if there is no such index.*

**Parameters**

value	- element to search for
-------	-------------------------

**Returns**

*the index of the first occurrence of the specified element in this list,*

**Exceptions**

NoSuchElementException	if value is not in the list
------------------------	-----------------------------

- virtual std::size\_t **lastIndexOf**(const E &value) throw ( decaf::lang::exceptions::NoSuchElementException )

*Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.*

*More formally, returns the highest index i such that get(i) == value or -1 if there is no such index.*

**Parameters**

value	- element to search for
-------	-------------------------

**Returns**

*the index of the last occurrence of the specified element in this list.*

**Exceptions**

NoSuchElementException	if value is not in the list
------------------------	-----------------------------

- virtual bool **isEmpty** () const

*Returns true if this collection contains no elements.*

*This implementation returns size() (p. 1226) == 0.*

**Returns**

*true if the size method return 0.*

- virtual std::size\_t **size** () const

*Returns the number of elements in this collection.*

*If this collection contains more than Integer.MAX\_VALUE elements, returns Integer.MAX\_VALUE.*

**Returns**

*the number of elements in this collection*

- virtual E **get** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Gets the element contained at position passed.*

**Parameters**

index	- position to get
-------	-------------------

**Returns**

*value at index*

- virtual E **set** (std::size\_t index, const E &element) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Replaces the element at the specified position in this list with the specified element.*

**Parameters**

index	- index of the element to replace
element	- element to be stored at the specified position

**Returns**

*the element previously at the specified position*

**Exceptions**

IndexOutOfBoundsException	- if the index is greater than size
---------------------------	-------------------------------------

- virtual bool **add** (const E &value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )

*Returns true if this collection changed as a result of the call.*

*(Returns false if this collection does not permit duplicates and already contains the specified element.)*

*Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 1216) classes should clearly specify in their documentation any restrictions on what elements may be added.*

*If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.*

*For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.*

**Parameters**

value	- reference to the element to add.
-------	------------------------------------

**Returns**

*true if the element was added*

**Exceptions**

UnsupportedOperationException	
IllegalArgumentException	
IllegalStateException	<i>if the element cannot be added at this time due to insertion restrictions</i>

- virtual void **add** (std::size\_t index, const E &element) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IndexOutOfBoundsException )

*Inserts the specified element at the specified position in this list.*

*Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).*

**Parameters**

index	- index at which the specified element is to be inserted
element	- element to be inserted

**Exceptions**

IndexOutOfBoundsException	- if the index is greater than size
UnsupportedOperationException	- If the collection is non-modifiable.

- virtual bool **addAll** (std::size\_t index, const **Collection**< E > &source) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Inserts all of the elements in the specified collection into this list at the specified position (optional operation).*

*Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)*

**Parameters**

index	The index at which to insert the first element from the specified collection
source	The <b>Collection</b> (p. 1216) containing elements to be added to this list

**Returns**

*true if this list changed as a result of the call*

**Exceptions**

IndexOutOfBoundsException	- if the index is greater than size
UnsupportedOperationException	- If the collection is non-modifiable.

- virtual bool **remove** (const E &value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )

*Removes a single instance of the specified element from this collection, if it is present (optional operation).*

*More formally, removes the first element *e* such that *e == o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).*

*This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.*

*Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.*

**Parameters**

value	- element to be removed from this collection, if present
-------	--

**Returns**

*true if an element was removed as a result of this call*

**Exceptions**

UnsupportedOperationException	<i>if the remove operation is not supported by this collection.</i>
IllegalArgumentException	<i>If the value is not a valid entry for this <b>Collection</b> (p. 1216).</i>

- virtual E **remove** (std::size\_t index) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Removes the element at the specified position in this list.*

*Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.*

**Parameters**

index	- the index of the element to be removed
-------	--

**Returns**

*the element previously at the specified position*

**Exceptions**

IndexOutOfBoundsException	- <i>if the index is greater than size</i>
UnsupportedOperationException	- <i>If the collection is non-modifiable.</i>

**6.772.1 Detailed Description**

`template<typename E> class decaf::util::StlList< E >`

**List** (p. 2409) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.

**6.772.2 Constructor & Destructor Documentation**

**6.772.2.1** `template<typename E> decaf::util::StlList< E >::StlList ( ) [inline]`

Default constructor - does nothing.

**6.772.2.2** `template<typename E> decaf::util::StlList< E >::StlList ( const StlList< E > & source ) [inline]`

Copy constructor - copies the content of the given set into this one.

**Parameters**

<i>source</i>	The source set.
---------------	-----------------

**6.772.2.3** `template<typename E> decaf::util::StlList< E >::StlList ( const Collection< E > & source ) [inline]`

Copy constructor - copies the content of the given set into this one.

**Parameters**

<i>source</i>	The source set.
---------------	-----------------

**6.772.2.4** `template<typename E> virtual decaf::util::StlList< E >::~~StlList ( ) [inline, virtual]`

**6.772.3 Member Function Documentation**

**6.772.3.1** `template<typename E> virtual bool decaf::util::StlList< E >::add ( const E & value ) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException ) [inline, virtual]`

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 1216) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

**Parameters**

<i>value</i>	- reference to the element to add.
--------------	------------------------------------

**Returns**

true if the element was added



**Exceptions**

<i>UnsupportedOperationException</i>	
<i>IllegalArgumentException</i>	
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions

Implements **decaf::util::Collection< E >** (p. 1218).

Referenced by decaf::util::StlList< cms::Connection \* >::addAll().

**6.772.3.2** `template<typename E> virtual void decaf::util::StlList< E >::add ( std::size_t index, const E & element ) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IndexOutOfBoundsException ) [inline, virtual]`

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

**Parameters**

<i>index</i>	- index at which the specified element is to be inserted
<i>element</i>	- element to be inserted

**Exceptions**

<i>IndexOutOfBoundsException</i>	- if the index is greater than size
<i>UnsupportedOperationException</i>	- If the collection is non-modifiable.

Implements **decaf::util::List< E >** (p. 2410).

**6.772.3.3** `template<typename E> virtual bool decaf::util::StlList< E >::addAll ( std::size_t index, const Collection< E > & source ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException ) [inline, virtual]`

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation

is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

#### Parameters

<i>index</i>	The index at which to insert the first element from the specified collection
<i>source</i>	The <b>Collection</b> (p. 1216) containing elements to be added to this list

#### Returns

true if this list changed as a result of the call

#### Exceptions

<i>IndexOutOfBoundsException</i>	- if the index is greater than size
<i>UnsupportedOperationException</i>	- If the collection is non-modifiable.

Implements **decaf::util::List< E >** (p. 2411).

**6.772.3.4** `template<typename E> virtual void decaf::util::StlList< E >::clear ( ) throw (lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 2223) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

#### Exceptions

<i>UnsupportedOperationException</i>	if the clear operation is not supported by this collection
--------------------------------------	--

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 163).

**6.772.3.5** `template<typename E> virtual bool decaf::util::StlList< E >::contains ( const E & value ) const throw ( lang::Exception ) [inline, virtual]`

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

**Parameters**

<i>value</i>	- the value whose presence is to be queried for in this <b>Collection</b> (p. 1216).
--------------	--

**Returns**

true if the value is contained in this collection

**Exceptions**

<i>Exception</i>	if an error occurs,
------------------	---------------------

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 164).

**6.772.3.6** `template<typename E> virtual void decaf::util::StlList< E >::copy ( const StlList< E > & source ) [inline, virtual]`

Referenced by `decaf::util::StlList< cms::Connection * >::StlList()`.

**6.772.3.7** `template<typename E> virtual bool decaf::util::StlList< E >::equals ( const StlList< E > & source ) const [inline, virtual]`

**6.772.3.8** `template<typename E> virtual E decaf::util::StlList< E >::get ( std::size_t index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [inline, virtual]`

Gets the element contained at position passed.

**Parameters**

<i>index</i>	- position to get
--------------	-------------------

**Returns**

value at index

Implements **decaf::util::List< E >** (p. 2412).

**6.772.3.9** `template<typename E> virtual std::size_t decaf::util::StlList< E >::indexOf ( const E & value ) throw ( decaf::lang::exceptions::NoSuchElementException ) [inline, virtual]`

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the lowest index *i* such that `get(i) == value`, or -1 if there is no such index.

**Parameters**

<i>value</i>	- element to search for
--------------	-------------------------

**Returns**

the index of the first occurrence of the specified element in this list,

**Exceptions**

<i>NoSuchElementException</i>	if value is not in the list
-------------------------------	-----------------------------

Implements **decaf::util::List< E >** (p. 2412).

**6.772.3.10** `template<typename E> virtual bool decaf::util::StlList< E >::isEmpty ( )  
const [inline, virtual]`

Returns true if this collection contains no elements.

This implementation returns **size()** (p. 1226) == 0.

**Returns**

true if the size method return 0.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 166).

**6.772.3.11** `template<typename E> virtual Iterator<E>* decaf::util::StlList< E  
>::iterator ( ) [inline, virtual]`

**Returns**

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p. 2221).

**6.772.3.12** `template<typename E> virtual Iterator<E>* decaf::util::StlList< E  
>::iterator ( ) const [inline, virtual]`

Implements **decaf::lang::Iterable< E >** (p. 2222).

**6.772.3.13** `template<typename E> virtual std::size_t decaf::util::StlList<  
E >::lastIndexOf ( const E & value ) throw (  
decaf::lang::exceptions::NoSuchElementException ) [inline,  
virtual]`

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the highest index i such that get(i) == value or -1 if there is no such index.

**Parameters**

<i>value</i>	- element to search for
--------------	-------------------------

**Returns**

the index of the last occurrence of the specified element in this list.

**Exceptions**

<i>NoSuchElementException</i>	if value is not in the list
-------------------------------	-----------------------------

Implements **decaf::util::List< E >** (p. 2413).

```
6.772.3.14  template<typename E> virtual ListIterator<E>*
             decaf::util::StlList< E >::listIterator ( std::size_t index ) throw (
             decaf::lang::exceptions::IndexOutOfBoundsException ) [inline,
             virtual]
```

**Parameters**

<i>index</i>	index of first element to be returned from the list iterator (by a call to the next method).
--------------	--

**Returns**

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if the index is out of range (index < 0    index > size() (p. 1226))
----------------------------------	--

Implements **decaf::util::List< E >** (p. 2415).

```
6.772.3.15  template<typename E> virtual ListIterator<E>* decaf::util::StlList< E
             >::listIterator ( ) [inline, virtual]
```

**Returns**

a list iterator over the elements in this list (in proper sequence).

Implements **decaf::util::List< E >** (p. 2414).

**6.772.3.16** `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator ( ) const [inline, virtual]`

Implements **decaf::util::List< E >** (p. 2414).

**6.772.3.17** `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator ( std::size_t index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [inline, virtual]`

Implements **decaf::util::List< E >** (p. 2414).

**6.772.3.18** `template<typename E> virtual E decaf::util::StlList< E >::remove ( std::size_t index ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException ) [inline, virtual]`

Removes the element at the specified position in this list.

Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

#### Parameters

<i>index</i>	- the index of the element to be removed
--------------	--

#### Returns

the element previously at the specified position

#### Exceptions

<i>IndexOutOfBoundsException</i>	- if the index is greater than size
<i>UnsupportedOperationException</i>	- If the collection is non-modifiable.

Implements **decaf::util::List< E >** (p. 2415).

**6.772.3.19** `template<typename E> virtual bool decaf::util::StlList< E >::remove ( const E & value ) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException ) [inline, virtual]`

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified

element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

#### Parameters

<i>value</i>	- element to be removed from this collection, if present
--------------	--

#### Returns

true if an element was removed as a result of this call

#### Exceptions

<i>UnsupportedOperationException</i>	if the remove operation is not supported by this collection.
<i>IllegalArgumentException</i>	If the value is not a valid entry for this <b>Collection</b> (p. 1216).

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 168).

```
6.772.3.20  template<typename E> virtual E decaf::util::StlList<
            E >::set ( std::size_t index, const E & element ) throw (
            decaf::lang::exceptions::IndexOutOfBoundsException ) [inline,
            virtual]
```

Replaces the element at the specified position in this list with the specified element.

#### Parameters

<i>index</i>	- index of the element to replace
<i>element</i>	- element to be stored at the specified position

#### Returns

the element previously at the specified position

#### Exceptions

<i>IndexOutOfBoundsException</i>	- if the index is greater than size
----------------------------------	-------------------------------------

Implements **decaf::util::List< E >** (p. 2416).

**6.772.3.21** `template<typename E> virtual std::size_t decaf::util::StlList< E >::size ( )`  
`const [inline, virtual]`

Returns the number of elements in this collection.

If this collection contains more than Integer.MAX\_VALUE elements, returns Integer.MAX\_VALUE.

### Returns

the number of elements in this collection

Implements **decaf::util::Collection< E >** (p. 1226).

Referenced by `decaf::util::StlList< cms::Connection * >::add()`, `decaf::util::StlList< cms::Connection * >::addAll()`, `decaf::util::StlList< cms::Connection * >::get()`, `decaf::util::StlList< cms::Connection * >::lastIndexOf()`, `decaf::util::StlList< cms::Connection * >::listIterator()`, `decaf::util::StlList< cms::Connection * >::remove()`, and `decaf::util::StlList< cms::Connection * >::set()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/StlList.h`

## 6.773 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference

**Map** (p. 2538) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

```
#include <src/main/decaf/util/StlMap.h>
```

Inheritance diagram for `decaf::util::StlMap< K, V, COMPARATOR >`:

### Public Member Functions

- **StlMap** ()  
*Default constructor - does nothing.*
- **StlMap** (const **StlMap** &source)  
*Copy constructor - copies the content of the given map into this one.*
- **StlMap** (const **Map**< K, V, COMPARATOR > &source)  
*Copy constructor - copies the content of the given map into this one.*
- virtual **~StlMap** ()
- virtual bool **equals** (const **StlMap** &source) const



## 6.773 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference 713

- virtual bool **equals** (const **Map**< K, V, COMPARATOR > &source) const

*Comparison, equality is dependent on the method of determining if the element are equal.*

### Parameters

source	- <b>Map</b> (p. 2538) to compare to this one.
--------	--

### Returns

*true if the **Map** (p. 2538) passed is equal in value to this one.*

- virtual void **copy** (const **StlMap** &source)

- virtual void **copy** (const **Map**< K, V, COMPARATOR > &source)

*Copies the content of the source map into this map.*

*Erases all existing data in this map.*

### Parameters

source	The source object to copy from.
--------	---------------------------------

- virtual void **clear** () throw ( decaf::lang::exceptions::UnsupportedOperationException )

*Removes all keys and values from this map.*

### Exceptions

UnsupportedOperationException	if this map is unmodifiable.
-------------------------------	------------------------------

- virtual bool **containsKey** (const K &key) const

*Indicates whether or this map contains a value for the given key.*

### Parameters

key	The key to look up.
-----	---------------------

### Returns

*true if this map contains the value, otherwise false.*

- virtual bool **containsValue** (const V &value) const

*Indicates whether or this map contains a value for the given value, i.e.*

*they are equal, this is done by operator== so the types must pass equivalence testing in this manner.*

### Parameters

value	The Value to look up.
-------	-----------------------

### Returns

*true if this map contains the value, otherwise false.*

- virtual bool **isEmpty** () const

**Returns**

if the **Map** (p. 2538) contains any element or not, *TRUE* or *FALSE*

- virtual std::size\_t **size** () const

**Returns**

The number of elements (key/value pairs) in this map.

- virtual V & **get** (const K &key) throw ( lang::exceptions::NoSuchElementException )

*Gets the value mapped to the specified key in the **Map** (p. 2538).*

*If there is no element in the map whose key is equivalent to the key provided then a *NoSuchElementException* is thrown.*

**Parameters**

key	The search key.
-----	-----------------

**Returns**

A reference to the value for the given key.

**Exceptions**

NoSuchElementException	if the key requests doesn't exist in the <b>Map</b> (p. 2538).
------------------------	--

- virtual const V & **get** (const K &key) const throw ( lang::exceptions::NoSuchElementException )

*Gets the value mapped to the specified key in the **Map** (p. 2538).*

*If there is no element in the map whose key is equivalent to the key provided then a *NoSuchElementException* is thrown.*

**Parameters**

key	The search key.
-----	-----------------

**Returns**

A {const} reference to the value for the given key.

**Exceptions**

NoSuchElementException	if the key requests doesn't exist in the <b>Map</b> (p. 2538).
------------------------	--

- virtual void **put** (const K &key, const V &value) throw ( decaf::lang::exceptions::UnsupportedOperationException )

*Sets the value for the specified key.*

**Parameters**

key	The target key.
value	The value to be set.

**Exceptions**

## 6.773 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference 1715

UnsupportedOperation Exception	<i>if this map is unmodifiable.</i>
-----------------------------------	-------------------------------------

- virtual void **putAll** (const **StlMap**< K, V, COMPARATOR > &other) throw ( decaf::lang::exceptions::UnsupportedOperationException )

- virtual void **putAll** (const **Map**< K, V, COMPARATOR > &other) throw ( decaf::lang::exceptions::UnsupportedOperationException )

*Stores a copy of the Mappings contained in the other **Map** (p. 2538) in this one.*

### Parameters

other	A <b>Map</b> (p. 2538) instance whose elements are to all be inserted in this <b>Map</b> (p. 2538).
-------	---

### Exceptions

UnsupportedOperation Exception	<i>If the implementing class does not support the putAll operation.</i>
-----------------------------------	---

- virtual V **remove** (const K &key) throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )

*Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.*

### Parameters

key	The search key.
-----	-----------------

### Returns

*a copy of the element that was previously mapped to the given key*

### Exceptions

NoSuchElementExcep- tion	<i>if this key is not in the <b>Map</b> (p. 2538).</i>
UnsupportedOperation Exception	<i>if this map is unmodifiable.</i>

- virtual std::vector< K > **keySet** () const

*Returns a **Set** (p. 3538) view of the mappings contained in this map.*

*The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 2223), **Set.remove** (p. 168), removeAll, retainAll and clear operations. It does not support the add or addAll operations.*

### Returns

*the entire set of keys in this map as a std::vector.*

- virtual std::vector< V > **values** () const

**Returns**

*the entire set of values in this map as a std::vector.*

- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )

*Locks the object.*

- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )

*Attempts to Lock the object, if the lock is already held by another thread than this method returns false.*

- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )

*Unlocks the object.*

- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals a waiter on this object that it can now wake up and continue.*

- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals the waiters on this object that it can now wake up and continue.*

**6.773.1 Detailed Description**

```
template<typename K, typename V, typename COMPARATOR = std::less<K>> class decaf::util::StlMap<
K, V, COMPARATOR >
```

**Map** (p. 2538) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

**Since**

1.0

## 6.773 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference 17

### 6.773.2 Constructor & Destructor Documentation

6.773.2.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
decaf::util::StlMap< K, V, COMPARATOR >::StlMap ( ) [inline]`

Default constructor - does nothing.

6.773.2.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
decaf::util::StlMap< K, V, COMPARATOR >::StlMap ( const StlMap< K, V,  
COMPARATOR > & source ) [inline]`

Copy constructor - copies the content of the given map into this one.

#### Parameters

<i>source</i>	The source map.
---------------	-----------------

6.773.2.3 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
decaf::util::StlMap< K, V, COMPARATOR >::StlMap ( const Map< K, V,  
COMPARATOR > & source ) [inline]`

Copy constructor - copies the content of the given map into this one.

#### Parameters

<i>source</i>	The source map.
---------------	-----------------

6.773.2.4 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual decaf::util::StlMap< K, V, COMPARATOR >::~~StlMap ( )  
[inline, virtual]`

### 6.773.3 Member Function Documentation

6.773.3.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual void decaf::util::StlMap< K, V, COMPARATOR >::clear ( ) throw  
( decaf::lang::exceptions::UnsupportedOperationException )  
[inline, virtual]`

Removes all keys and values from this map.

#### Exceptions

<i>UnsupportedOperation Exception</i>	if this map is unmodifiable.
---	------------------------------

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2540).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::copy()`.

```
6.773.3.2  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::StlMap< K, V, COMPARATOR >::containsKey ( const K &
            key ) const    [inline, virtual]
```

Indicates whether or this map contains a value for the given key.

#### Parameters

<i>key</i>	The key to look up.
------------	---------------------

#### Returns

true if this map contains the value, otherwise false.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2541).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::equals()`.

```
6.773.3.3  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::StlMap< K, V, COMPARATOR >::containsValue ( const V
            & value ) const    [inline, virtual]
```

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

#### Parameters

<i>value</i>	The Value to look up.
--------------	-----------------------

#### Returns

true if this map contains the value, otherwise false.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2542).

```
6.773.3.4  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual void decaf::util::StlMap< K, V, COMPARATOR >::copy ( const StlMap<
            K, V, COMPARATOR > & source )    [inline, virtual]
```

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::StlMap()`.

```
6.773.3.5  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual void decaf::util::StlMap< K, V, COMPARATOR >::copy ( const Map< K,
            V, COMPARATOR > & source )    [inline, virtual]
```

Copies the content of the source map into this map.

### 6.773 `decaf::util::StlMap< K, V, COMPARATOR >` Class Template Reference 19

Erases all existing data in this map.

#### Parameters

<i>source</i>	The source object to copy from.
---------------	---------------------------------

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2543).

**6.773.3.6** `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual bool decaf::util::StlMap< K, V, COMPARATOR >::equals ( const Map<  
K, V, COMPARATOR > & source ) const [inline, virtual]`

Comparison, equality is dependent on the method of determining if the element are equal.

#### Parameters

<i>source</i>	- <b>Map</b> (p. 2538) to compare to this one.
---------------	--

#### Returns

true if the **Map** (p. 2538) passed is equal in value to this one.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2543).

**6.773.3.7** `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual bool decaf::util::StlMap< K, V, COMPARATOR >::equals ( const  
StlMap< K, V, COMPARATOR > & source ) const [inline, virtual]`

**6.773.3.8** `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual V& decaf::util::StlMap< K, V, COMPARATOR >::get ( const K & key )  
throw ( lang::exceptions::NoSuchElementException ) [inline,  
virtual]`

Gets the value mapped to the specified key in the **Map** (p. 2538).

If there is no element in the map whose key is equivalent to the key provided then a `NoSuchElementException` is thrown.

#### Parameters

<i>key</i>	The search key.
------------	-----------------

#### Returns

A reference to the value for the given key.

#### Exceptions

<i>NoSuchElementException</i>	if the key requests doesn't exist in the <b>Map</b> (p. 2538).
-------------------------------	--

Implements **decaf::util::Map**< **K**, **V**, **COMPARATOR** > (p. 2543).

```
6.773.3.9  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual const V& decaf::util::StlMap< K, V, COMPARATOR >::get ( const K & key
            ) const throw ( lang::exceptions::NoSuchElementException )  [inline,
            virtual]
```

Gets the value mapped to the specified key in the **Map** (p. 2538).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** is thrown.

#### Parameters

<i>key</i>	The search key.
------------	-----------------

#### Returns

A {const} reference to the value for the given key.

#### Exceptions

<i>NoSuchElementException</i>	if the key requests doesn't exist in the <b>Map</b> (p. 2538).
-------------------------------	--

Implements **decaf::util::Map**< **K**, **V**, **COMPARATOR** > (p. 2544).

```
6.773.3.10  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::StlMap< K, V, COMPARATOR >::isEmpty ( ) const
            [inline, virtual]
```

#### Returns

if the **Map** (p. 2538) contains any element or not, TRUE or FALSE

Implements **decaf::util::Map**< **K**, **V**, **COMPARATOR** > (p. 2545).

```
6.773.3.11  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual std::vector<K> decaf::util::StlMap< K, V, COMPARATOR >::keySet ( )
            const  [inline, virtual]
```

Returns a **Set** (p. 3538) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 2223), **Set.remove** (p. 168), **removeAll**, **retainAll** and **clear** operations. It does not support the **add** or **addAll** operations.



## 6.773 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference 3721

### Returns

the entire set of keys in this map as a std::vector.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2546).

**6.773.3.12** `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual void decaf::util::StlMap< K, V, COMPARATOR >::lock ( ) throw (  
decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Locks the object.

### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3812).

**6.773.3.13** `template<typename K, typename V, typename COMPARATOR =  
std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR  
>::notify ( ) throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalMonitorStateException) [inline,  
virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3813).

**6.773.3.14** `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual void decaf::util::StlMap< K, V, COMPARATOR >::notifyAll  
( ) throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalMonitorStateException) [inline,  
virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
-------------------------------------	--

<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.
-------------------------	---

Implements **decaf::util::concurrent::Synchronizable** (p. 3814).

```
6.773.3.15  template<typename K, typename V, typename COMPARTOR
= std::less<K>> virtual void decaf::util::StlMap< K, V,
COMPARTOR >::put ( const K & key, const V & value ) throw (
decaf::lang::exceptions::UnsupportedOperationException )
[inline, virtual]
```

Sets the value for the specified key.

#### Parameters

<i>key</i>	The target key.
<i>value</i>	The value to be set.

#### Exceptions

<i>UnsupportedOperationException</i>	if this map is unmodifiable.
--------------------------------------	------------------------------

Implements **decaf::util::Map< K, V, COMPARTOR >** (p. 2547).

Referenced by **decaf::util::StlMap< std::string, cms::Topic \* >::putAll()**.

```
6.773.3.16  template<typename K, typename V, typename COMPARTOR =
std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARTOR
>::putAll ( const StlMap< K, V, COMPARTOR > & other ) throw (
decaf::lang::exceptions::UnsupportedOperationException )
[inline, virtual]
```

Referenced by **decaf::util::StlMap< std::string, cms::Topic \* >::copy()**.

```
6.773.3.17  template<typename K, typename V, typename COMPARTOR =
std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARTOR
>::putAll ( const Map< K, V, COMPARTOR > & other ) throw (
decaf::lang::exceptions::UnsupportedOperationException )
[inline, virtual]
```

Stores a copy of the Mappings contained in the other **Map** (p. 2538) in this one.

#### Parameters

<i>other</i>	A <b>Map</b> (p. 2538) instance whose elements are to all be inserted in this <b>Map</b> (p. 2538).
--------------	---

## 6.773 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference 723

### Exceptions

<i>UnsupportedOperationException</i>	If the implementing class does not support the putAll operation.
--------------------------------------	--

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2548).

**6.773.3.18** `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual V decaf::util::StlMap< K, V, COMPARATOR >::remove ( const K &  
key ) throw ( decaf::lang::exceptions::NoSuchElementException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[inline, virtual]`

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

### Parameters

<i>key</i>	The search key.
------------	-----------------

### Returns

a copy of the element that was previously mapped to the given key

### Exceptions

<i>NoSuchElementException</i>	if this key is not in the <b>Map</b> (p. 2538).
<i>UnsupportedOperationException</i>	if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2548).

**6.773.3.19** `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual std::size_t decaf::util::StlMap< K, V, COMPARATOR >::size ( ) const  
[inline, virtual]`

### Returns

The number of elements (key/value pairs) in this map.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2549).

**6.773.3.20** `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual bool decaf::util::StlMap< K, V, COMPARATOR >::tryLock ( ) throw (   
decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

**Returns**

true if the lock was acquired, false if it is already held by another thread.

**Exceptions**

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3815).

```
6.773.3.21  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual void decaf::util::StlMap< K, V, COMPARATOR >::unlock ( ) throw (
            decaf::lang::exceptions::RuntimeException ) [inline, virtual]
```

Unlocks the object.

**Exceptions**

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3816).

```
6.773.3.22  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual std::vector<V> decaf::util::StlMap< K, V, COMPARATOR >::values ( )
            const [inline, virtual]
```

**Returns**

the entire set of values in this map as a std::vector.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2550).

Referenced by decaf::util::StlMap< std::string, cms::Topic \* >::values().

```
6.773.3.23  template<typename K, typename V, typename COMPARATOR =
            std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR
            >::wait ( ) throw ( decaf::lang::exceptions::RuntimeException,
            decaf::lang::exceptions::IllegalMonitorStateException,
            decaf::lang::exceptions::InterruptedException ) [inline,
            virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

**Exceptions**

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

## 6.773 `decaf::util::StlMap< K, V, COMPARATOR >` Class Template Reference 3725

Implements `decaf::util::concurrent::Synchronizable` (p. 3818).

```
6.773.3.24  template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::StlMap< K, V, COMPARATOR >::wait ( long long
millisecs, int nanos ) throw ( decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException ) [inline,
virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

### Exceptions

<i>IllegalArgumentException</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3820).

```
6.773.3.25  template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::StlMap< K, V, COMPARATOR >::wait ( long
long millisecs ) throw ( decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException ) [inline,
virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

**Parameters**

<i>milliseconds</i>	the time in milliseconds to wait, or WAIT_INFINITE
---------------------	--

**Exceptions**

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3819).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/StlMap.h`

**6.774 decaf::util::StlQueue< T > Class Template Reference**

The **Queue** (p. 3239) class accepts messages with an `psuh(m)` command where `m` is the message to be queued.

```
#include <src/main/decaf/util/StlQueue.h>
```

Inheritance diagram for `decaf::util::StlQueue< T >`:

**Data Structures**

- class **QueueIterator**

**Public Member Functions**

- **StlQueue ()**
- virtual **~StlQueue ()**
- **Iterator< T > \* iterator ()**  
*Gets an **Iterator** (p. 2222) over this **Queue** (p. 3239).*
- void **clear ()**  
*Empties this queue.*
- T & **front ()**  
*Returns a Reference to the element at the head of the queue.*
- const T & **front () const**  
*Returns a Reference to the element at the head of the queue.*

- **T & back ()**  
*Returns a Reference to the element at the tail of the queue.*
- **const T & back () const**  
*Returns a Reference to the element at the tail of the queue.*
- **void push (const T &t)**  
*Places a new Object at the Tail of the queue.*
- **void enqueueFront (const T &t)**  
*Places a new Object at the front of the queue.*
- **T pop ()**  
*Removes and returns the element that is at the Head of the queue.*
- **size\_t size () const**  
*Gets the Number of elements currently in the **Queue** (p. 3239).*
- **bool empty () const**  
*Checks if this **Queue** (p. 3239) is currently empty.*
- **virtual std::vector< T > toArray () const**
- **void reverse (StlQueue< T > &target) const**  
*Reverses the order of the contents of this queue and stores them in the target queue.*
- **virtual void lock () throw ( decaf::lang::exceptions::RuntimeException )**  
*Locks the object.*
- **virtual bool tryLock () throw ( decaf::lang::exceptions::RuntimeException )**  
*Attempts to Lock the object, if the lock is already held by another thread than this method returns false.*
- **virtual void unlock () throw ( decaf::lang::exceptions::RuntimeException )**  
*Unlocks the object.*
- **virtual void wait () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )**  
*Waits on a signal from this object, which is generated by a call to Notify.*
- **virtual void wait (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )**  
*Waits on a signal from this object, which is generated by a call to Notify.*
- **virtual void wait (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )**

*Waits on a signal from this object, which is generated by a call to `Notify`.*

- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals a waiter on this object that it can now wake up and continue.*

- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals the waiters on this object that it can now wake up and continue.*

- T & **getSafeValue** ()

*Fetch a reference to the safe value this object will return when there is nothing to fetch from the queue.*

### 6.774.1 Detailed Description

```
template<typename T> class decaf::util::StlQueue< T >
```

The **Queue** (p. 3239) class accepts messages with an `push(m)` command where `m` is the message to be queued. It destructively returns the message with `pop()` (p. 3728). `pop()` (p. 3728) returns messages in the order they were enqueued.

**Queue** (p. 3239) is implemented with an instance of the STL queue object. The interface is essentially the same as that of the STL queue except that the `pop` method actually reaturns a reference to the element popped. This frees the app from having to call the `front` method before calling `pop`.

```
Queue<string> sq; // make a queue to hold string messages
sq.push(s); // enqueues a message m
string s = sq.pop(); // dequeues a message
```

#### = DESIGN CONSIDERATIONS

The **Queue** (p. 3239) class inherits from the `Synchronizable` interface and provides methods for locking and unlocking this queue as well as waiting on this queue. In a multi-threaded app this can allow for multiple threads to be reading from and writing to the same **Queue** (p. 3239).

Clients should consider that in a multiple threaded app it is possible that items could be placed on the queue faster than you are taking them off, so protection should be placed in your polling loop to ensure that you don't get stuck there.



## 6.774.2 Constructor & Destructor Documentation

6.774.2.1 `template<typename T> decaf::util::StdQueue< T >::StdQueue ( )`  
[inline]

6.774.2.2 `template<typename T> virtual decaf::util::StdQueue< T >::~~StdQueue ( )`  
[inline, virtual]

## 6.774.3 Member Function Documentation

6.774.3.1 `template<typename T> T& decaf::util::StdQueue< T >::back ( )`  
[inline]

Returns a Reference to the element at the tail of the queue.

### Returns

reference to a queue type object or (safe)

6.774.3.2 `template<typename T> const T& decaf::util::StdQueue< T >::back ( ) const`  
[inline]

Returns a Reference to the element at the tail of the queue.

### Returns

reference to a queue type object or (safe)

6.774.3.3 `template<typename T> void decaf::util::StdQueue< T >::clear ( )`  
[inline]

Empties this queue.

6.774.3.4 `template<typename T> bool decaf::util::StdQueue< T >::empty ( ) const`  
[inline]

Checks if this **Queue** (p. 3239) is currently empty.

### Returns

boolean indicating queue emptiness

6.774.3.5 `template<typename T> void decaf::util::StdQueue< T >::enqueueFront ( const T & t )` [inline]

Places a new Object at the front of the queue.

**Parameters**

<i>t</i>	- <b>Queue</b> (p. 3239) Object Type reference.
----------	---

**6.774.3.6** `template<typename T> const T& decaf::util::StlQueue< T >::front ( ) const`  
`[inline]`

Returns a Reference to the element at the head of the queue.

**Returns**

reference to a queue type object or (safe)

**6.774.3.7** `template<typename T> T& decaf::util::StlQueue< T >::front ( )`  
`[inline]`

Returns a Reference to the element at the head of the queue.

**Returns**

reference to a queue type object or (safe)

**6.774.3.8** `template<typename T> T& decaf::util::StlQueue< T >::getSafeValue ( )`  
`[inline]`

Fetch a reference to the safe value this object will return when there is nothing to fetch from the queue.

**Returns**

Reference to this Queues safe object

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::back()`, `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::front()`, and `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::pop()`.

**6.774.3.9** `template<typename T> Iterator<T>* decaf::util::StlQueue< T >::iterator ( )`  
`[inline]`

Gets an **Iterator** (p. 2222) over this **Queue** (p. 3239).

**Returns**

new iterator pointer that is owned by the caller.

```
6.774.3.10  template<typename T> virtual void decaf::util::StlQueue< T >::lock ( )
            throw ( decaf::lang::exceptions::RuntimeException ) [inline,
            virtual]
```

Locks the object.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3812).

```
6.774.3.11  template<typename T> virtual void decaf::util::StlQueue< T
            >::notify ( ) throw ( decaf::lang::exceptions::RuntimeException,
            decaf::lang::exceptions::IllegalMonitorStateException ) [inline,
            virtual]
```

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

#### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3813).

```
6.774.3.12  template<typename T> virtual void decaf::util::StlQueue< T
            >::notifyAll ( ) throw ( decaf::lang::exceptions::RuntimeException,
            decaf::lang::exceptions::IllegalMonitorStateException ) [inline,
            virtual]
```

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

#### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3814).

**6.774.3.13** `template<typename T> T decaf::util::StlQueue< T >::pop ( ) [inline]`

Removes and returns the element that is at the Head of the queue.

#### Returns

reference to a queue type object or (safe)

**6.774.3.14** `template<typename T> void decaf::util::StlQueue< T >::push ( const T & t ) [inline]`

Places a new Object at the Tail of the queue.

#### Parameters

<i>t</i>	- <b>Queue</b> (p. 3239) Object Type reference.
----------	---

**6.774.3.15** `template<typename T> void decaf::util::StlQueue< T >::reverse ( StlQueue< T > & target ) const [inline]`

Reverses the order of the contents of this queue and stores them in the target queue.

#### Parameters

<i>target</i>	- The target queue that will receive the contents of this queue in reverse order.
---------------	---

**6.774.3.16** `template<typename T> size_t decaf::util::StlQueue< T >::size ( ) const [inline]`

Gets the Number of elements currently in the **Queue** (p. 3239).

#### Returns

**Queue** (p. 3239) Size

**6.774.3.17** `template<typename T> virtual std::vector<T> decaf::util::StlQueue< T >::toArray ( ) const [inline, virtual]`

#### Returns

the all values in this queue as a std::vector.

**6.774.3.18** `template<typename T> virtual bool decaf::util::StlQueue< T >::tryLock (`  
`) throw ( decaf::lang::exceptions::RuntimeException ) [inline,`  
`virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

#### Returns

true if the lock was acquired, false if it is already held by another thread.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3815).

**6.774.3.19** `template<typename T> virtual void decaf::util::StlQueue< T >::unlock (`  
`) throw ( decaf::lang::exceptions::RuntimeException ) [inline,`  
`virtual]`

Unlocks the object.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3816).

**6.774.3.20** `template<typename T> virtual void decaf::util::StlQueue< T`  
`>::wait ( ) throw ( decaf::lang::exceptions::RuntimeException,`  
`decaf::lang::exceptions::IllegalMonitorStateException,`  
`decaf::lang::exceptions::InterruptedException ) [inline,`  
`virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorState-Exception</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3818).

**6.774.3.21** `template<typename T> virtual void decaf::util::StlQueue< T >::wait ( long long millisecs ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3819).

**6.774.3.22** `template<typename T> virtual void decaf::util::StlQueue< T >::wait ( long long millisecs, int nanos ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

#### Exceptions

<i>IllegalArgumentException</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
---------------------------------	--

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorState-Exception</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3820).

The documentation for this class was generated from the following file:

- src/main/decaf/util/StlQueue.h

## 6.775 decaf::util::StlSet< E > Class Template Reference

**Set** (p. 3538) template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set.

```
#include <src/main/decaf/util/StlSet.h>
```

Inheritance diagram for decaf::util::StlSet< E >:

### Data Structures

- class **ConstSetIterator**
- class **SetIterator**

### Public Member Functions

- **StlSet** ()  
*Default constructor - does nothing.*
- **StlSet** (const **StlSet** &source)  
*Copy constructor - copies the content of the given set into this one.*
- **StlSet** (const **Collection**< E > &source)  
*Copy constructor - copies the content of the given set into this one.*
- virtual **~StlSet** ()
- **Iterator**< E > \* **iterator** ()  
*Returns*  
*an iterator over a set of elements of type T.*
- **Iterator**< E > \* **iterator** () const
- virtual bool **equals** (const **StlSet** &source) const

- virtual void **copy** (const **StdSet** &source)
- virtual void **clear** () throw ( lang::exceptions::UnsupportedOperationException )

*Removes all of the elements from this collection (optional operation).*

*The collection will be empty after this method returns.*

*This implementation iterates over this collection, removing each element using the **Iterator.remove** (p.2223) operation. Most implementations will probably choose to override this method for efficiency.*

*Note that this implementation will throw an **UnsupportedOperationException** if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.*

#### **Exceptions**

UnsupportedOperationException	if the clear operation is not supported by this collection
-------------------------------	--

- virtual bool **contains** (const E &value) const throw ( lang::Exception )

*Returns true if this collection contains the specified element.*

*This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.*

#### **Parameters**

value	- the value whose presence is to be queried for in this <b>Collection</b> (p.1216).
-------	---

#### **Returns**

*true if the value is contained in this collection*

#### **Exceptions**

Exception	if an error occurs,
-----------	---------------------

- virtual bool **isEmpty** () const
- virtual std::size\_t **size** () const
- virtual bool **add** (const E &value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )

*Returns true if this collection changed as a result of the call.*

*(Returns false if this collection does not permit duplicates and already contains the specified element.)*

*Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p.1216) classes should clearly specify in their documentation any restrictions on what elements may be added.*

*If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.*



For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

**Parameters**

value	- reference to the element to add.
-------	------------------------------------

**Returns**

*true if the element was added*

**Exceptions**

UnsupportedOperationException	
IllegalArgumentException	
IllegalStateException	<i>if the element cannot be added at this time due to insertion restrictions</i>

- virtual bool **remove** (const E &value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )

*Removes a single instance of the specified element from this collection, if it is present (optional operation).*

*More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).*

*This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.*

*Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.*

**Parameters**

value	- element to be removed from this collection, if present
-------	--

**Returns**

*true if an element was removed as a result of this call*

**Exceptions**

UnsupportedOperationException	<i>if the remove operation is not supported by this collection.</i>
IllegalArgumentException	<i>If the value is not a valid entry for this <b>Collection</b> (p. 1216).</i>

**6.775.1 Detailed Description**

template<typename E> class decaf::util::StlSet< E >

**Set** (p. 3538) template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set.

## 6.775.2 Constructor & Destructor Documentation

6.775.2.1 `template<typename E> decaf::util::StlSet< E >::StlSet ( ) [inline]`

Default constructor - does nothing.

6.775.2.2 `template<typename E> decaf::util::StlSet< E >::StlSet ( const StlSet< E > & source ) [inline]`

Copy constructor - copies the content of the given set into this one.

### Parameters

<i>source</i>	The source set.
---------------	-----------------

6.775.2.3 `template<typename E> decaf::util::StlSet< E >::StlSet ( const Collection< E > & source ) [inline]`

Copy constructor - copies the content of the given set into this one.

### Parameters

<i>source</i>	The source set.
---------------	-----------------

6.775.2.4 `template<typename E> virtual decaf::util::StlSet< E >::~~StlSet ( ) [inline, virtual]`

## 6.775.3 Member Function Documentation

6.775.3.1 `template<typename E> virtual bool decaf::util::StlSet< E >::add ( const E & value ) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException ) [inline, virtual]`

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 1216) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after

this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

#### Parameters

<i>value</i>	- reference to the element to add.
--------------	------------------------------------

#### Returns

true if the element was added

#### Exceptions

<i>UnsupportedOperationException</i>	
<i>IllegalArgumentException</i>	
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions

Implements **decaf::util::Collection< E >** (p. 1218).

**6.775.3.2** `template<typename E> virtual void decaf::util::StdSet< E >::clear ( ) throw ( lang::exceptions::UnsupportedOperationException ) [inline, virtual]`

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 2223) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

#### Exceptions

<i>UnsupportedOperationException</i>	if the clear operation is not supported by this collection
--------------------------------------	--

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 163).

**6.775.3.3** `template<typename E> virtual bool decaf::util::StlSet< E >::contains ( const E & value ) const throw ( lang::Exception ) [inline, virtual]`

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

#### Parameters

<i>value</i>	- the value whose presence is to be queried for in this <b>Collection</b> (p. 1216).
--------------	--

#### Returns

true if the value is contained in this collection

#### Exceptions

<i>Exception</i>	if an error occurs,
------------------	---------------------

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 164).

**6.775.3.4** `template<typename E> virtual void decaf::util::StlSet< E >::copy ( const StlSet< E > & source ) [inline, virtual]`

Referenced by `decaf::util::StlSet< ActiveMQSession * >::StlSet()`.

**6.775.3.5** `template<typename E> virtual bool decaf::util::StlSet< E >::equals ( const StlSet< E > & source ) const [inline, virtual]`

**6.775.3.6** `template<typename E> virtual bool decaf::util::StlSet< E >::isEmpty ( ) const [inline, virtual]`

#### Returns

if the set contains any element or not, TRUE or FALSE

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 166).

**6.775.3.7** `template<typename E> Iterator<E>* decaf::util::StlSet< E >::iterator ( ) const [inline, virtual]`

Implements **decaf::lang::Iterable< E >** (p. 2222).

**6.775.3.8** `template<typename E> Iterator<E>* decaf::util::StlSet< E >::iterator ( ) [inline, virtual]`

#### Returns

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p. 2221).

**6.775.3.9** `template<typename E> virtual bool decaf::util::StlSet< E >::remove ( const E  
& value ) throw ( lang::exceptions::UnsupportedOperationException,  
lang::exceptions::IllegalArgumentException ) [inline, virtual]`

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

#### Parameters

<i>value</i>	- element to be removed from this collection, if present
--------------	--

#### Returns

true if an element was removed as a result of this call

#### Exceptions

<i>UnsupportedOperationException</i>	if the remove operation is not supported by this collection.
<i>IllegalArgumentException</i>	If the value is not a valid entry for this <b>Collection</b> (p. 1216).

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 168).

**6.775.3.10** `template<typename E> virtual std::size_t decaf::util::StlSet< E >::size ( )  
const [inline, virtual]`

#### Returns

The number of elements in this set.

Implements **decaf::util::Collection< E >** (p. 1226).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/StlSet.h`

## 6.776 activemq::wireformat::stomp::StompCommandConstants Class Reference

```
#include <src/main/activemq/wireformat/stomp/StompCommandConstants.h>
```

### Static Public Attributes

- static const std::string **CONNECT**
- static const std::string **CONNECTED**
- static const std::string **DISCONNECT**
- static const std::string **SUBSCRIBE**
- static const std::string **UNSUBSCRIBE**
- static const std::string **MESSAGE**
- static const std::string **SEND**
- static const std::string **BEGIN**
- static const std::string **COMMIT**
- static const std::string **ABORT**
- static const std::string **ACK**
- static const std::string **ERROR\_CMD**
- static const std::string **RECEIPT**
- static const std::string **HEADER\_DESTINATION**
- static const std::string **HEADER\_TRANSACTIONID**
- static const std::string **HEADER\_CONTENTLENGTH**
- static const std::string **HEADER\_SESSIONID**
- static const std::string **HEADER\_RECEIPT\_REQUIRED**
- static const std::string **HEADER\_RECEIPTID**
- static const std::string **HEADER\_MESSAGEID**
- static const std::string **HEADER\_ACK**
- static const std::string **HEADER\_LOGIN**
- static const std::string **HEADER\_PASSWORD**
- static const std::string **HEADER\_CLIENT\_ID**
- static const std::string **HEADER\_MESSAGE**
- static const std::string **HEADER\_CORRELATIONID**
- static const std::string **HEADER\_REQUESTID**
- static const std::string **HEADER\_RESPONSEID**
- static const std::string **HEADER\_EXPIRES**
- static const std::string **HEADER\_PERSISTENT**
- static const std::string **HEADER\_REPLYTO**
- static const std::string **HEADER\_TYPE**
- static const std::string **HEADER\_DISPATCH\_ASYNC**
- static const std::string **HEADER\_EXCLUSIVE**
- static const std::string **HEADER\_MAXPENDINGMSGLIMIT**
- static const std::string **HEADER\_NOLOCAL**
- static const std::string **HEADER\_PREFETCHSIZE**
- static const std::string **HEADER\_JMSPRIORITY**

- static const std::string **HEADER\_CONSUMERPRIORITY**
- static const std::string **HEADER\_RETROACTIVE**
- static const std::string **HEADER\_SUBSCRIPTIONNAME**
- static const std::string **HEADER\_OLDSUBSCRIPTIONNAME**
- static const std::string **HEADER\_TIMESTAMP**
- static const std::string **HEADER\_REDELIVERED**
- static const std::string **HEADER\_REDELIVERYCOUNT**
- static const std::string **HEADER\_SELECTOR**
- static const std::string **HEADER\_ID**
- static const std::string **HEADER\_SUBSCRIPTION**
- static const std::string **HEADER\_TRANSFORMATION**
- static const std::string **HEADER\_TRANSFORMATION\_ERROR**
- static const std::string **ACK\_CLIENT**
- static const std::string **ACK\_AUTO**
- static const std::string **ACK\_INDIVIDUAL**
- static const std::string **TEXT**
- static const std::string **BYTES**
- static const std::string **QUEUE\_PREFIX**
- static const std::string **TOPIC\_PREFIX**
- static const std::string **TEMPQUEUE\_PREFIX**
- static const std::string **TEMPTOPIC\_PREFIX**

**6.776.1 Field Documentation**

- 6.776.1.1 **const std::string activemq::wireformat::stomp::StompCommandConstants::ABORT**  
[static]
- 6.776.1.2 **const std::string activemq::wireformat::stomp::StompCommandConstants::ACK**  
[static]
- 6.776.1.3 **const std::string activemq::wireformat::stomp::StompCommandConstants::ACK\_-  
AUTO** [static]
- 6.776.1.4 **const std::string activemq::wireformat::stomp::StompCommandConstants::ACK\_-  
CLIENT** [static]
- 6.776.1.5 **const std::string activemq::wireformat::stomp::StompCommandConstants::ACK\_-  
INDIVIDUAL** [static]
- 6.776.1.6 **const std::string activemq::wireformat::stomp::StompCommandConstants::BEGIN**  
[static]
- 6.776.1.7 **const std::string activemq::wireformat::stomp::StompCommandConstants::BYTES**  
[static]
- 6.776.1.8 **const std::string activemq::wireformat::stomp::StompCommandConstants::COMMIT**  
[static]
- 6.776.1.9 **const std::string activemq::wireformat::stomp::StompCommandConstants::CONNECT**  
[static]
- 6.776.1.10 **const std::string activemq::wireformat::stomp::StompCommandConstants::CONNECTED**  
[static]
- 6.776.1.11 **const std::string activemq::wireformat::stomp::StompCommandConstants::DISCONNECT**  
[static]
- 6.776.1.12 **const std::string activemq::wireformat::stomp::StompCommandConstants::ERROR\_-  
CMD** [static]
- 6.776.1.13 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER\_-  
ACK** [static]
- 6.776.1.14 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER\_-  
CLIENT\_ID** [static]
- 6.776.1.15 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER\_-  
CONSUMERPRIORITY** [static]
- 6.776.1.16 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER\_-  
CONTENTLENGTH** [static]
- 6.776.1.17 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER\_-  
CORRELATIONID** [static]
- 6.776.1.18 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER\_-  
DESTINATION** [static]
- 6.776.1.19 **const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER\_-  
DISPATCH\_ASYNC** [static]



- src/main/activemq/wireformat/stomp/StompCommandConstants.h

## 6.777 activemq::wireformat::stomp::StompFrame Class Reference

A Stomp-level message frame that encloses all messages to and from the broker.

```
#include <src/main/activemq/wireformat/stomp/StompFrame.h>
```

### Public Member Functions

- **StompFrame ()**  
*Default constructor.*
- virtual **~StompFrame ()**  
*Destruction.*
- **StompFrame \* clone () const**  
*Clonse this message exactly, returns a new instance that the caller is required to delete.*
- void **copy** (const **StompFrame** \*src)  
*Copies the contents of the passed Frame to this one.*
- void **setCommand** (const std::string &cmd)  
*Sets the command for this stomp frame.*
- const std::string & **getCommand () const**  
*Accessor for this frame's command field.*
- bool **hasProperty** (const std::string &name) const  
*Checks if the given property is present in the Frame.*
- std::string **getProperty** (const std::string &name, const std::string &fallback="") const  
*Gets a property from this Frame's properties and returns it, or the default value given.*
- std::string **removeProperty** (const std::string &name)  
*Gets and remove the property specified, if the property is not set, this method returns the empty string.*
- void **setProperty** (const std::string &name, const std::string &value)  
*Sets the property given to the value specified in this Frame's Properties.*
- **decaf::util::Properties & getProperties ()**  
*Gets access to the header properties for this frame.*

- const **decaf::util::Properties** & **getProperties** () const
- const std::vector< unsigned char > & **getBody** () const  
*Accessor for the body data of this frame.*
- std::vector< unsigned char > & **getBody** ()  
*Non-const version of the body accessor.*
- std::size\_t **getBodyLength** () const  
*Return the number of bytes contained in this frames body.*
- void **setBody** (const unsigned char \*bytes, std::size\_t numBytes)  
*Sets the body data of this frame as a byte sequence.*
- void **toStream** (**decaf::io::DataOutputStream** \*stream) const throw ( decaf::io::IOException )  
*Writes this Frame to an OuputStream in the Stomp Wire Format.*
- void **fromStream** (**decaf::io::DataInputStream** \*stream) throw ( decaf::io::IOException )  
*Reads a Stop Frame from a DataInputStream in the Stomp Wire format.*

### 6.777.1 Detailed Description

A Stomp-level message frame that encloses all messages to and from the broker.

### 6.777.2 Constructor & Destructor Documentation

#### 6.777.2.1 **activemq::wireformat::stomp::StompFrame::StompFrame ( )** [inline]

Default constructor.

#### 6.777.2.2 **virtual activemq::wireformat::stomp::StompFrame::~~StompFrame ( )** [inline, virtual]

Destruction.

### 6.777.3 Member Function Documentation

#### 6.777.3.1 **StompFrame\*** **activemq::wireformat::stomp::StompFrame::clone ( )** const

Clone this message exactly, returns a new instance that the caller is required to delete.

**Returns**

new copy of this message

**6.777.3.2 void activemq::wireformat::stomp::StompFrame::copy ( const StompFrame \* *src* )**

Copies the contents of the passed Frame to this one.

**Parameters**

<i>src</i>	- Frame to copy
------------	-----------------

**6.777.3.3 void activemq::wireformat::stomp::StompFrame::fromStream ( decaf::io::DataInputStream \* *stream* ) throw ( decaf::io::IOException )**

Reads a Stop Frame from a DataInputStream in the Stomp Wire format.

**Parameters**

<i>stream</i>	- The stream to read the Frame from.
---------------	--------------------------------------

**Exceptions**

<i>IOException</i>	if an error occurs while writing the Frame.
--------------------	---

**6.777.3.4 const std::vector<unsigned char>& activemq::wireformat::stomp::StompFrame::getBody ( ) const [inline]**

Accessor for the body data of this frame.

**Returns**

char pointer to body data

**6.777.3.5 std::vector<unsigned char>& activemq::wireformat::stomp::StompFrame::getBody ( ) [inline]**

Non-const version of the body accessor.

**6.777.3.6 std::size\_t activemq::wireformat::stomp::StompFrame::getBodyLength ( ) const [inline]**

Return the number of bytes contained in this frames body.

**Returns**

Body bytes length.

**6.777.3.7** `const std::string& activemq::wireformat::stomp::StompFrame::getCommand ( )`  
`const [inline]`

Accessor for this frame's command field.

**6.777.3.8** `decaf::util::Properties& activemq::wireformat::stomp::StompFrame::getProperties ( )` `[inline]`

Gets access to the header properties for this frame.

**Returns**

the Properties object owned by this Frame

**6.777.3.9** `const decaf::util::Properties& activemq::wireformat::stomp::StompFrame::getProperties ( )`  
`const [inline]`

**6.777.3.10** `std::string activemq::wireformat::stomp::StompFrame::getProperty ( const std::string & name, const std::string & fallback = " " ) const` `[inline]`

Gets a property from this Frame's properties and returns it, or the default value given.

**Parameters**

<i>name</i>	- The name of the property to lookup
<i>fallback</i>	- The default value to return if this value isn't set

**Returns**

string value of the property asked for.

**6.777.3.11** `bool activemq::wireformat::stomp::StompFrame::hasProperty ( const std::string & name ) const` `[inline]`

Checks if the given property is present in the Frame.

**Parameters**

<i>name</i>	- The name of the property to check for.
-------------	--

**6.777.3.12** `std::string activemq::wireformat::stomp::StompFrame::removeProperty ( const std::string & name ) [inline]`

Gets and remove the property specified, if the property is not set, this method returns the empty string.

#### Parameters

<i>name</i>	- the Name of the property to get and return.
-------------	---

**6.777.3.13** `void activemq::wireformat::stomp::StompFrame::setBody ( const unsigned char * bytes, std::size_t numBytes )`

Sets the body data of this frame as a byte sequence.

#### Parameters

<i>bytes</i>	The byte buffer to be set in the body.
<i>numBytes</i>	The number of bytes in the buffer.

**6.777.3.14** `void activemq::wireformat::stomp::StompFrame::setCommand ( const std::string & cmd ) [inline]`

Sets the command for this stomp frame.

#### Parameters

<i>cmd</i>	command The command to be set.
------------	--------------------------------

**6.777.3.15** `void activemq::wireformat::stomp::StompFrame::setProperty ( const std::string & name, const std::string & value ) [inline]`

Sets the property given to the value specified in this Frame's Properties.

#### Parameters

<i>name</i>	- Name of the property.
<i>value</i>	- Value to set the property to.

**6.777.3.16** `void activemq::wireformat::stomp::StompFrame::toStream ( decaf::io::DataOutputStream * stream ) const throw ( decaf::io::IOException )`

Writes this Frame to an OuputStream in the Stomp Wire Format.

**Parameters**

<i>stream</i>	- The stream to write the Frame to.
---------------	-------------------------------------

**Exceptions**

<i>IOException</i>	if an error occurs while reading the Frame.
--------------------	---

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/stomp/**StompFrame.h**

**6.778 activemq::wireformat::stomp::StompHelper Class Reference**

Utility Methods used when marshaling to and from StompFrame's.

```
#include <src/main/activemq/wireformat/stomp/StompHelper.h>
```

**Public Member Functions**

- **StompHelper** ()
- virtual **~StompHelper** ()
- void **convertProperties** (const **Pointer**< **StompFrame** > &frame, const **Pointer**< **Message** > &message)  
*Converts the Headers in a Stomp Frame into Headers in the given Message Command.*
- void **convertProperties** (const **Pointer**< **Message** > &message, const **Pointer**< **StompFrame** > &frame)  
*Converts the Properties in a Message Command to Valid Headers and Properties in the **StompFrame** (p. 3741).*
- **Pointer**< **ActiveMQDestination** > **convertDestination** (const std::string &destination)  
*Converts from a Stomp Destination to an ActiveMQDestination.*
- std::string **convertDestination** (const **Pointer**< **ActiveMQDestination** > &destination)  
*Converts from a ActiveMQDestination to a Stomp Destination Name.*
- std::string **convertMessageId** (const **Pointer**< **MessageId** > &messageId)  
*Converts a MessageId instance to a Stomp MessageId String.*
- **Pointer**< **MessageId** > **convertMessageId** (const std::string &messageId)  
*Converts a Stomp MessageId string to a MessageId.*
- std::string **convertConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)

*Converts a ConsumerId instance to a Stomp ConsumerId String.*

- **Pointer< ConsumerId > convertConsumerId** (const std::string &consumerId)

*Converts a Stomp ConsumerId string to a ConsumerId.*

- std::string **convertProducerId** (const **Pointer< ProducerId >** &producerId)

*Converts a ProducerId instance to a Stomp ProducerId String.*

- **Pointer< ProducerId > convertProducerId** (const std::string &producerId)

*Converts a Stomp ProducerId string to a ProducerId.*

- std::string **convertTransactionId** (const **Pointer< TransactionId >** &transactionId)

*Converts a TransactionId instance to a Stomp TransactionId String.*

- **Pointer< TransactionId > convertTransactionId** (const std::string &transactionId)

*Converts a Stomp TransactionId string to a TransactionId.*

### 6.778.1 Detailed Description

Utility Methods used when marshaling to and from StompFrame's.

Since

3.0

### 6.778.2 Constructor & Destructor Documentation

6.778.2.1 **activemq::wireformat::stomp::StompHelper::StompHelper** ( ) `[inline]`

6.778.2.2 **virtual activemq::wireformat::stomp::StompHelper::~~StompHelper** ( )  
`[inline, virtual]`

### 6.778.3 Member Function Documentation

6.778.3.1 **std::string activemq::wireformat::stomp::StompHelper::convertConsumerId** ( const **Pointer< ConsumerId >** & *consumerId* )

Converts a ConsumerId instance to a Stomp ConsumerId String.

#### Parameters

<i>consumerId</i>	- the Consumer instance to convert.
-------------------	-------------------------------------

**Returns**

a Stomp Consumer Id String.

**6.778.3.2 `Pointer<ConsumerId> activemq::wireformat::stomp::StompHelper::convertConsumerId ( const std::string & consumerId )`**

Converts a Stomp ConsumerId string to a ConsumerId.

**Parameters**

<i>consumerId</i>	- the String Consumer Id to convert.
-------------------	--------------------------------------

**Returns**

Pointer to a new ConsumerId.

**6.778.3.3 `std::string activemq::wireformat::stomp::StompHelper::convertDestination ( const Pointer<ActiveMQDestination> & destination )`**

Converts from a ActiveMQDestination to a Stomp Destination Name.

**Parameters**

<i>destination</i>	- The ActiveMQDestination to Convert
--------------------	--------------------------------------

**Returns**

the Stomp String name that defines the destination.

**6.778.3.4 `Pointer<ActiveMQDestination> activemq::wireformat::stomp::StompHelper::convertDestination ( const std::string & destination )`**

Converts from a Stomp Destination to an ActiveMQDestination.

**Parameters**

<i>destination</i>	- The Stomp Destination name string.
--------------------	--------------------------------------

**Returns**

Pointer to a new ActiveMQDestination.

**6.778.3.5 `std::string activemq::wireformat::stomp::StompHelper::convertMessageId ( const Pointer<MessageId> & messageId )`**

Converts a MessageId instance to a Stomp MessageId String.



**Parameters**

<i>messageId</i>	- the MessageId instance to convert.
------------------	--------------------------------------

**Returns**

a Stomp Message Id String.

**6.778.3.6** `Pointer<MessageId> activemq::wireformat::stomp::StompHelper::convertMessageId ( const std::string & messageId )`

Converts a Stomp MessageId string to a MessageId.

**Parameters**

<i>messageId</i>	- the String message Id to convert.
------------------	-------------------------------------

**Returns**

Pointer to a new MessageId.

**6.778.3.7** `std::string activemq::wireformat::stomp::StompHelper::convertProducerId ( const Pointer< ProducerId > & producerId )`

Converts a ProducerId instance to a Stomp ProducerId String.

**Parameters**

<i>producerId</i>	- the Producer instance to convert.
-------------------	-------------------------------------

**Returns**

a Stomp Producer Id String.

**6.778.3.8** `Pointer<ProducerId> activemq::wireformat::stomp::StompHelper::convertProducerId ( const std::string & producerId )`

Converts a Stomp ProducerId string to a ProducerId.

**Parameters**

<i>producerId</i>	- the String Producer Id to convert.
-------------------	--------------------------------------

**Returns**

Pointer to a new ProducerId.

**6.778.3.9** `void activemq::wireformat::stomp::StompHelper::convertProperties ( const  
Pointer< Message > & message, const Pointer< StompFrame > & frame )`

Converts the Properties in a Message Command to Valid Headers and Properties in the **StompFrame** (p. 3741).

#### Parameters

<i>message</i>	- The message to move the Headers to.
<i>frame</i>	- The frame to extract headers from.

**6.778.3.10** `void activemq::wireformat::stomp::StompHelper::convertProperties ( const  
Pointer< StompFrame > & frame, const Pointer< Message > & message )`

Converts the Headers in a Stomp Frame into Headers in the given Message Command.

#### Parameters

<i>frame</i>	- The frame to extract headers from.
<i>message</i>	- The message to move the Headers to.

**6.778.3.11** `Pointer<TransactionId> ac-  
tivemq::wireformat::stomp::StompHelper::convertTransactionId (   
const std::string & transactionId )`

Converts a Stomp TransactionId string to a TransactionId.

#### Parameters

<i>transac- tionId</i>	- the String Transaction Id to convert.
----------------------------	---

#### Returns

Pointer to a new TransactionId.

**6.778.3.12** `std::string activemq::wireformat::stomp::StompHelper::convertTransactionId ( const  
Pointer< TransactionId > & transactionId )`

Converts a TransactionId instance to a Stomp TransactionId String.

#### Parameters

<i>transac- tionId</i>	- the Transaction instance to convert.
----------------------------	--

**Returns**

a Stomp Transaction Id String.

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/stomp/StompHelper.h`

## 6.779 `activemq::wireformat::stomp::StompWireFormat` Class Reference

```
#include <src/main/activemq/wireformat/stomp/StompWireFormat.h>
```

Inheritance diagram for `activemq::wireformat::stomp::StompWireFormat`:

**Public Member Functions**

- **StompWireFormat** ()
- virtual **~StompWireFormat** ()
- virtual void **marshal** (const **Pointer**< **commands::Command** > &command, const **activemq::transport::Transport** \*transport, **decaf::io::DataOutputStream** \*out) throw ( **decaf::io::IOException** )  
*Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.*
- virtual **Pointer**< **commands::Command** > **unmarshal** (const **activemq::transport::Transport** \*transport, **decaf::io::DataInputStream** \*in) throw ( **decaf::io::IOException** )  
*Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.*
- virtual void **setVersion** (int version AMQCPP\_UNUSED)  
*Set the Version.*
- virtual int **getVersion** () const  
*Get the Version.*
- virtual bool **inReceive** () const  
*Is there a Message being unmarshaled?*
- virtual bool **hasNegotiator** () const  
*Returns true if this WireFormat (p.4088) has a Negotiator that needs to wrap the Transport that uses it.*

- virtual **Pointer**< **transport::Transport** > **createNegotiator** (const **Pointer**< **transport::Transport** > &transport) throw ( decaf::lang::exceptions::UnsupportedOperationException )

*If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.*

## 6.779.1 Constructor & Destructor Documentation

6.779.1.1 **activemq::wireformat::stomp::StompWireFormat::StompWireFormat** ( )

6.779.1.2 virtual **activemq::wireformat::stomp::StompWireFormat::~~StompWireFormat** ( )  
[virtual]

## 6.779.2 Member Function Documentation

6.779.2.1 virtual **Pointer**<transport::Transport>  
**activemq::wireformat::stomp::StompWireFormat::createNegotiator** (  
const **Pointer**< **transport::Transport** > & *transport* ) throw (  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]

If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

### Returns

new instance of a **WireFormatNegotiator** (p. 4129).

### Exceptions

<i>UnsupportedOperation Exception</i>	if the <b>WireFormat</b> (p. 4088) doesn't have a Negotiator.
---	---

Implements **activemq::wireformat::WireFormat** (p. 4089).

6.779.2.2 virtual int **activemq::wireformat::stomp::StompWireFormat::getVersion** ( ) const  
[inline, virtual]

Get the Version.

### Returns

the version of the wire format

Implements **activemq::wireformat::WireFormat** (p. 4090).

**6.779.2.3** `virtual bool activemq::wireformat::stomp::StompWireFormat::hasNegotiator ( ) const`  
`[inline, virtual]`

Returns true if this **WireFormat** (p. 4088) has a Negotiator that needs to wrap the Transport that uses it.

#### Returns

true if the **WireFormat** (p. 4088) provides a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 4090).

**6.779.2.4** `virtual bool activemq::wireformat::stomp::StompWireFormat::inReceive ( ) const`  
`[inline, virtual]`

Is there a Message being unmarshaled?

#### Returns

true while in the doUnmarshal method.

Implements **activemq::wireformat::WireFormat** (p. 4090).

**6.779.2.5** `virtual void activemq::wireformat::stomp::StompWireFormat::marshal`  
`( const Pointer< commands::Command > & command,`  
`const activemq::transport::Transport * transport,`  
`decaf::io::DataOutputStream * out ) throw ( decaf::io::IOException )`  
`[virtual]`

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

#### Parameters

<i>command</i>	The Command to Marshal to the output stream.
<i>transport</i>	The Transport that initiated this marshal call.
<i>out</i>	The output stream to write the command to.

#### Exceptions

<i>IOException</i>	
--------------------	--

Implements **activemq::wireformat::WireFormat** (p. 4091).

**6.779.2.6** `virtual void activemq::wireformat::stomp::StompWireFormat::setVersion ( int version`  
`AMQCPP_UNUSED ) [inline, virtual]`

Set the Version.

**Parameters**

<i>the</i>	version of the wire format
------------	----------------------------

Implements **activemq::wireformat::WireFormat** (p. 4091).

**6.779.2.7** `virtual Pointer<commands::Command>`  
`activemq::wireformat::stomp::StompWireFormat::unmarshal ( const`  
`activemq::transport::Transport * transport, decaf::io::DataInputStream`  
`* in ) throw ( decaf::io::IOException ) [virtual]`

Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.

Returns a Pointer to the newly unmarshaled Command.

**Parameters**

<i>transport</i>	- Pointer to the transport that is making this request.
<i>in</i>	- the input stream to read the command from.

**Returns**

the newly marshaled Command, caller owns the pointer

**Exceptions**

<i>IOException</i>
--------------------

Implements **activemq::wireformat::WireFormat** (p. 4091).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/stomp/StompWireFormat.h`

## 6.780 **activemq::wireformat::stomp::StompWireFormatFactory Class Reference**

Factory used to create the Stomp Wire Format instance.

```
#include <src/main/activemq/wireformat/stomp/StompWireFormatFactory.h>
```

Inheritance diagram for `activemq::wireformat::stomp::StompWireFormatFactory`:

**Public Member Functions**

- `StompWireFormatFactory ()`

- virtual `~StompWireFormatFactory()`
- virtual `Pointer<WireFormat> createWireFormat(const decaf::util::Properties &properties) throw ( decaf::lang::exceptions::IllegalStateException )`

*Creates a new **WireFormat** (p. 4088) Object passing it a set of properties from which it can obtain any optional settings.*

### 6.780.1 Detailed Description

Factory used to create the Stomp Wire Format instance.

### 6.780.2 Constructor & Destructor Documentation

**6.780.2.1** `activemq::wireformat::stomp::StompWireFormatFactory::StompWireFormatFactory()` `[inline]`

**6.780.2.2** `virtual activemq::wireformat::stomp::StompWireFormatFactory::~~StompWireFormatFactory()` `[inline, virtual]`

### 6.780.3 Member Function Documentation

**6.780.3.1** `virtual Pointer<WireFormat> activemq::wireformat::stomp::StompWireFormatFactory::createWireFormat(const decaf::util::Properties & properties) throw ( decaf::lang::exceptions::IllegalStateException )` `[virtual]`

Creates a new **WireFormat** (p. 4088) Object passing it a set of properties from which it can obtain any optional settings.

#### Parameters

<i>properties</i>	- the Properties for this <b>WireFormat</b> (p. 4088)
-------------------	---

Implements `activemq::wireformat::WireFormatFactory` (p. 4093).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/stomp/StompWireFormatFactory.h`

## 6.781 cms::Stoppable Class Reference

Interface for a class that implements the stop method.

```
#include <src/main/cms/Stoppable.h>
```

Inheritance diagram for cms::Stoppable:

## Public Member Functions

- virtual `~Stoppable()`
- virtual void `stop()`=0 throw ( `CMSEException` )

*Stops this service.*

### 6.781.1 Detailed Description

Interface for a class that implements the stop method. An object that implements this interface implies that it will halt all operations that result in events being propagated to external users, internally the Object can continue to process data but not events will be generated to clients and methods that return data will not return valid results until the object is started again.

#### Since

1.0

### 6.781.2 Constructor & Destructor Documentation

6.781.2.1 virtual `cms::Stoppable::~~Stoppable()` [`inline`, `virtual`]

### 6.781.3 Member Function Documentation

6.781.3.1 virtual void `cms::Stoppable::stop()` throw ( `CMSEException` ) [`pure virtual`]

Stops this service.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if an internal error occurs while stopping the Service.
--	---

Implemented in `activemq::core::ActiveMQConnection` (p. 280).

The documentation for this class was generated from the following file:

- `src/main/cms/Stoppable.h`

## 6.782 decaf::util::logging::StreamHandler Class Reference

Stream based logging **Handler** (p. 2042).

```
#include <src/main/decaf/util/logging/StreamHandler.h>
```



Inheritance diagram for decaf::util::logging::StreamHandler:

## Public Member Functions

- **StreamHandler** ()  
*Create a **StreamHandler** (p. 3756), with no current output stream.*
- **StreamHandler** (decaf::io::OutputStream \*stream, Formatter \*formatter)  
*Create a **StreamHandler** (p. 3756), with no current output stream.*
- virtual ~**StreamHandler** ()
- virtual void **close** () throw ( decaf::io::IOException )  
*Close the current output stream.*
- virtual void **flush** ()  
*Flush the Handler's output, clears any buffers.*
- virtual void **publish** (const **LogRecord** &record)  
*Publish the Log Record to this **Handler** (p. 2042).*
- virtual bool **isLoggable** (const **LogRecord** &record) const  
*Check if this **Handler** (p. 2042) would actually log a given **LogRecord** (p. 2487).*

## Protected Member Functions

- virtual void **setOutputStream** (decaf::io::OutputStream \*stream) throw ( decaf::lang::exceptions::NullPointerException )  
*Change the output stream.*
- void **close** (bool closeStream)  
*Closes this handler, but the underlying output stream is only closed if closeStream is true.*

### 6.782.1 Detailed Description

Stream based logging **Handler** (p. 2042). This is primarily intended as a base class or support class to be used in implementing other logging Handlers.

LogRecords are published to a given **decaf::io::OutputStream** (p. 2991).

Configuration: By default each **StreamHandler** (p. 3756) is initialized using the following **LogManager** (p. 2480) configuration properties. If properties are not defined (or have invalid values) then the specified default values are used.

\* `decaf.util.logging.StreamHandler.level` specifies the default level for the **Handler** (p. 2042) (defaults to **Level.INFO** (p. 2408)). \* `decaf.util.logging.StreamHandler.filter` specifies the name of a **Filter** (p. 1949) class to use (defaults to no **Filter** (p. 1949)). \* `decaf.util.logging.StreamHandler.formatter` specifies the name of a **Formatter** (p. 2027) class to use (defaults to **decaf.util.logging.SimpleFormatter** (p. 3604)).

### Since

1.0

## 6.782.2 Constructor & Destructor Documentation

### 6.782.2.1 `decaf::util::logging::StreamHandler::StreamHandler ( )`

Create a **StreamHandler** (p. 3756), with no current output stream.

### 6.782.2.2 `decaf::util::logging::StreamHandler::StreamHandler ( decaf::io::OutputStream * stream, Formatter * formatter )`

Create a **StreamHandler** (p. 3756), with no current output stream.

### 6.782.2.3 `virtual decaf::util::logging::StreamHandler::~StreamHandler ( )` [virtual]

## 6.782.3 Member Function Documentation

### 6.782.3.1 `virtual void decaf::util::logging::StreamHandler::close ( ) throw ( decaf::io::IOException )` [virtual]

Close the current output stream.

The close method will perform a flush and then close the **Handler** (p. 2042). After close has been called this **Handler** (p. 2042) should no longer be used. Method calls may either be silently ignored or may throw runtime exceptions.

### Exceptions

<i>IOException</i> if an I/O error occurs.
--

Implements **decaf::io::Closeable** (p. 1181).

Reimplemented in **decaf::util::logging::ConsoleHandler** (p. 1440).

### 6.782.3.2 `void decaf::util::logging::StreamHandler::close ( bool closeStream )` [protected]

Closes this handler, but the underlying output stream is only closed if `closeStream` is true.

**Parameters**

<i>closeStream</i>	whether to close the underlying output stream.
--------------------	--

**6.782.3.3** virtual void decaf::util::logging::StreamHandler::flush ( ) [virtual]

Flush the Handler's output, clears any buffers.

Implements **decaf::util::logging::Handler** (p. 2043).

**6.782.3.4** virtual bool decaf::util::logging::StreamHandler::isLoggable ( const **LogRecord** & *record* ) const [virtual]

Check if this **Handler** (p. 2042) would actually log a given **LogRecord** (p. 2487).

**Parameters**

<i>record</i>	<b>LogRecord</b> (p. 2487) to check
---------------	-------------------------------------

**Returns**

true if the record can be logged with current settings.

Reimplemented from **decaf::util::logging::Handler** (p. 2044).

**6.782.3.5** virtual void decaf::util::logging::StreamHandler::publish ( const **LogRecord** & *record* ) [virtual]

Publish the Log Record to this **Handler** (p. 2042).

**Parameters**

<i>record</i>	The <b>LogRecord</b> (p. 2487) to Publish
---------------	---

Implements **decaf::util::logging::Handler** (p. 2044).

Reimplemented in **decaf::util::logging::ConsoleHandler** (p. 1441).

**6.782.3.6** virtual void decaf::util::logging::StreamHandler::setOutputStream ( decaf::io::OutputStream \* *stream* ) throw ( decaf::lang::exceptions::NullPointerException ) [protected, virtual]

Change the output stream.

If there is a current output stream then the Formatter's tail string is written and the stream is flushed and closed. Then the output stream is replaced with the new output stream.

**Parameters**

<i>stream</i>	The new output stream. May not be NULL.
---------------	---

**Exceptions**

<i>NullPointerException</i>	if the passed stream is NULL.
-----------------------------	-------------------------------

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**StreamHandler.h**

**6.783 cms::StreamMessage Class Reference**

Interface for a **StreamMessage** (p. 3760).

```
#include <src/main/cms/StreamMessage.h>
```

Inheritance diagram for cms::StreamMessage:

**Public Member Functions**

- virtual **~StreamMessage** ()
- virtual bool **readBoolean** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Boolean from the Stream message stream.*
- virtual void **writeBoolean** (bool value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a boolean to the Stream message stream as a 1-byte value.*
- virtual unsigned char **readByte** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Byte from the Stream message stream.*
- virtual void **writeByte** (unsigned char value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a byte to the Stream message stream as a 1-byte value.*
- virtual int **readBytes** (std::vector< unsigned char > &value) const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a byte array from the Stream message stream.*

- virtual void **writeBytes** (const std::vector< unsigned char > &value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.*
- virtual int **readBytes** (unsigned char \*buffer, int length) const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a portion of the Stream message stream.*
- virtual void **writeBytes** (const unsigned char \*value, int offset, int length)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a portion of a byte array to the Stream message stream.*
- virtual char **readChar** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Char from the Stream message stream.*
- virtual void **writeChar** (char value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a char to the Stream message stream as a 1-byte value.*
- virtual float **readFloat** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 32 bit float from the Stream message stream.*
- virtual void **writeFloat** (float value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a float to the Stream message stream as a 4 byte value.*
- virtual double **readDouble** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 64 bit double from the Stream message stream.*
- virtual void **writeDouble** (double value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a double to the Stream message stream as a 8 byte value.*
- virtual short **readShort** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 16 bit signed short from the Stream message stream.*
- virtual void **writeShort** (short value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a signed short to the Stream message stream as a 2 byte value.*

- virtual unsigned short **readUnsignedShort** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 16 bit unsigned short from the Stream message stream.*
- virtual void **writeUnsignedShort** (unsigned short value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a unsigned short to the Stream message stream as a 2 byte value.*
- virtual int **readInt** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 32 bit signed integer from the Stream message stream.*
- virtual void **writeInt** (int value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a signed int to the Stream message stream as a 4 byte value.*
- virtual long long **readLong** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 64 bit long from the Stream message stream.*
- virtual void **writeLong** (long long value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a long long to the Stream message stream as a 8 byte value.*
- virtual std::string **readString** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads an ASCII String from the Stream message stream.*
- virtual void **writeString** (const std::string &value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes an ASCII String to the Stream message stream.*

### 6.783.1 Detailed Description

Interface for a **StreamMessage** (p. 3760). The stream Messages provides a **Message** (p. 2614) type whose body is a stream of self describing primitive types. The primitive values are read and written using accessors specific to the given types.

**StreamMessage** (p. 3760) objects support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSEException** (p. 1190). The string-to- primitive conversions may throw a runtime exception if the primitive's `valueOf()` method does not accept it as a valid String representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	char	int	long	float	double	String	byte[]
boolean	X									X
byte		X	X		X	X				X
short			X		X	X				X
char				X						X
int					X	X				X
long						X				X
float							X	X		X
double								X		X
String	X	X	X		X	X	X	X	X	
byte[]										X

Since

1.3

## 6.783.2 Constructor & Destructor Documentation

**6.783.2.1** virtual cms::StreamMessage::~StreamMessage ( ) [inline, virtual]

## 6.783.3 Member Function Documentation

**6.783.3.1** virtual bool cms::StreamMessage::readBoolean ( ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]

Reads a Boolean from the Stream message stream.

### Returns

boolean value from stream

### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of message stream has been reached.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 541).

**6.783.3.2** `virtual unsigned char cms::StreamMessage::readByte ( ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads a Byte from the Stream message stream.

### Returns

unsigned char value from stream

### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of message stream has been reached.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 541).

**6.783.3.3** `virtual int cms::StreamMessage::readBytes ( std::vector< unsigned char > & value ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads a byte array from the Stream message stream.

If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

### Parameters

<i>value</i>	buffer to place data in
--------------	-------------------------

### Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached



**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of message stream has been reached.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 542).

**6.783.3.4** `virtual int cms::StreamMessage::readBytes ( unsigned char * buffer, int length ) const  
throw ( cms::MessageEOFException, cms::MessageFormatException,  
cms::MessageNotReadableException, cms::CMSException ) [pure  
virtual]`

Reads a portion of the Stream message stream.

If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an **CMSException** (p. 1190) is thrown. No bytes will be read from the stream for this exception case.

**Parameters**

<i>buffer</i>	the buffer into which the data is read
<i>length</i>	the number of bytes to read; must be less than or equal to value.length

**Returns**

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of message stream has been reached.

<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 542).

**6.783.3.5** `virtual char cms::StreamMessage::readChar ( ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )` [pure virtual]

Reads a Char from the Stream message stream.

#### Returns

char value from stream

#### Exceptions

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of message stream has been reached.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 543).

**6.783.3.6** `virtual double cms::StreamMessage::readDouble ( ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )` [pure virtual]

Reads a 64 bit double from the Stream message stream.

#### Returns

double value from stream

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of message stream has been reached.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 544).

**6.783.3.7** `virtual float cms::StreamMessage::readFloat ( ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads a 32 bit float from the Stream message stream.

**Returns**

double value from stream

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of message stream has been reached.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 544).

**6.783.3.8** `virtual int cms::StreamMessage::readInt ( ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads a 32 bit signed integer from the Stream message stream.

**Returns**

int value from stream

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of message stream has been reached.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 545).

**6.783.3.9** virtual long long cms::StreamMessage::readLong ( ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]

Reads a 64 bit long from the Stream message stream.

**Returns**

long long value from stream

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of message stream has been reached.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 545).

**6.783.3.10** virtual short cms::StreamMessage::readShort ( ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]

Reads a 16 bit signed short from the Stream message stream.

#### Returns

short value from stream

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of message stream has been reached.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 546).

**6.783.3.11** virtual std::string cms::StreamMessage::readString ( ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]

Reads an ASCII String from the Stream message stream.

#### Returns

String from stream

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of message stream has been reached.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 546).

**6.783.3.12** `virtual unsigned short cms::StreamMessage::readUnsignedShort ( ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads a 16 bit unsigned short from the Stream message stream.

### Returns

unsigned short value from stream

### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2747)	- if unexpected end of message stream has been reached.
<b><i>MessageFormatException</i></b> (p. 2749)	- if this type conversion is invalid.
<b><i>MessageNotReadableException</i></b> (p. 2807)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 547).

**6.783.3.13** `virtual void cms::StreamMessage::writeBoolean ( bool value ) throw ( cms::MessageNotWritableException, cms::CMSException ) [pure virtual]`

Writes a boolean to the Stream message stream as a 1-byte value.

The value true is written as the value (byte)1; the value false is written as the value (byte)0.

### Parameters

<i>value</i>	boolean to write to the stream
--------------	--------------------------------

### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWritableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 548).

**6.783.3.14** `virtual void cms::StreamMessage::writeByte ( unsigned char value ) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Writes a byte to the Stream message stream as a 1-byte value.

#### Parameters

<i>value</i>	byte to write to the stream
--------------	-----------------------------

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 548).

**6.783.3.15** `virtual void cms::StreamMessage::writeBytes ( const unsigned char * value, int offset, int length ) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Writes a portion of a byte array to the Stream message stream.

size as the number of bytes to write.

#### Parameters

<i>value</i>	bytes to write to the stream
<i>offset</i>	the initial offset within the byte array
<i>length</i>	the number of bytes to use

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 548).

**6.783.3.16** `virtual void cms::StreamMessage::writeBytes ( const std::vector< unsigned char > & value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.

#### Parameters

<i>value</i>	bytes to write to the stream
--------------	------------------------------

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 549).

**6.783.3.17** `virtual void cms::StreamMessage::writeChar ( char value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a char to the Stream message stream as a 1-byte value.

#### Parameters

<i>value</i>	char to write to the stream
--------------	-----------------------------

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 549).

**6.783.3.18** `virtual void cms::StreamMessage::writeDouble ( double value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a double to the Stream message stream as a 8 byte value.

#### Parameters

<i>value</i>	double to write to the stream
--------------	-------------------------------



**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 550).

**6.783.3.19** `virtual void cms::StreamMessage::writeFloat ( float value ) throw ( cms::MessageNotWriteableException, cms::CMSEException )` [pure virtual]

Writes a float to the Stream message stream as a 4 byte value.

**Parameters**

<i>value</i>	float to write to the stream
--------------	------------------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 550).

**6.783.3.20** `virtual void cms::StreamMessage::writeInt ( int value ) throw ( cms::MessageNotWriteableException, cms::CMSEException )` [pure virtual]

Writes a signed int to the Stream message stream as a 4 byte value.

**Parameters**

<i>value</i>	signed int to write to the stream
--------------	-----------------------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 551).

**6.783.3.21** `virtual void cms::StreamMessage::writeLong ( long long value ) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Writes a long long to the Stream message stream as a 8 byte value.

#### Parameters

<i>value</i>	signed long long to write to the stream
--------------	---

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 551).

**6.783.3.22** `virtual void cms::StreamMessage::writeShort ( short value ) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Writes a signed short to the Stream message stream as a 2 byte value.

#### Parameters

<i>value</i>	signed short to write to the stream
--------------	-------------------------------------

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 551).

**6.783.3.23** `virtual void cms::StreamMessage::writeString ( const std::string & value ) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Writes an ASCII String to the Stream message stream.

#### Parameters

<i>value</i>	String to write to the stream
--------------	-------------------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 552).

**6.783.3.24** `virtual void cms::StreamMessage::writeUnsignedShort ( unsigned short value )  
throw ( cms::MessageNotWriteableException, cms::CMSEException )  
[pure virtual]`

Writes a unsigned short to the Stream message stream as a 2 byte value.

**Parameters**

<i>value</i>	unsigned short to write to the stream
--------------	---------------------------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 552).

The documentation for this class was generated from the following file:

- `src/main/cms/StreamMessage.h`

**6.784 decaf::lang::String Class Reference**

The **String** (p. 3775) class represents an immutable sequence of chars.

```
#include <src/main/decaf/lang/String.h>
```

Inheritance diagram for decaf::lang::String:

**Public Member Functions**

- **String ()**  
*Creates a new empty **String** (p. 3775) object.*

- **String** (const std::string &source)

Create a new **String** (p. 3775) object that represents the given STL string.

- virtual ~**String** ()
- bool **isEmpty** () const
- virtual int **length** () const

#### Returns

the length of the underlying character sequence.

- virtual char **charAt** (int index) const throw ( lang::exceptions::IndexOutOfBoundsException )

Returns the Char at the specified index so long as the index is not greater than the length of the sequence.

#### Parameters

index	- position to return the char at.
-------	-----------------------------------

#### Returns

the char at the given position

#### Exceptions

IndexOutOfBoundsException	if index is > than <b>length</b> () (p. 1168) or negative
---------------------------	---

- virtual **CharSequence \* subSequence** (int start, int end) const throw ( lang::exceptions::IndexOutOfBoundsException )

Returns a new **CharSequence** (p. 1167) that is a subsequence of this sequence. The subsequence starts with the char value at the specified index and ends with the char value at index end - 1. The length (in chars) of the returned sequence is end - start, so if start == end then an empty sequence is returned.

#### Parameters

start	- the start index, inclusive
end	- the end index, exclusive

#### Returns

a new **CharSequence** (p. 1167)

#### Exceptions

IndexOutOfBoundsException	if start or end > <b>length</b> () (p. 1168) or start or end are negative.
---------------------------	--

- virtual std::string **toString** () const

#### Returns

the string representation of this **CharSequence** (p. 1167)

### 6.784.1 Detailed Description

The **String** (p. 3775) class represents an immutable sequence of chars.

#### Since

1.0

### 6.784.2 Constructor & Destructor Documentation

#### 6.784.2.1 decaf::lang::String::String ( )

Creates a new empty **String** (p. 3775) object.

#### 6.784.2.2 decaf::lang::String::String ( const std::string & source )

Create a new **String** (p. 3775) object that represents the given STL string.

#### Parameters

<i>source</i>	The string to copy into this new <b>String</b> (p. 3775) object.
---------------	--

#### 6.784.2.3 virtual decaf::lang::String::~~String ( ) [virtual]

### 6.784.3 Member Function Documentation

#### 6.784.3.1 virtual char decaf::lang::String::charAt ( int *index* ) const throw ( lang::exceptions::IndexOutOfBoundsException ) [virtual]

Returns the Char at the specified index so long as the index is not greater than the length of the sequence.

#### Parameters

<i>index</i>	- position to return the char at.
--------------	-----------------------------------

#### Returns

the char at the given position

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is > than <b>length()</b> (p. 1168) or negative
----------------------------------	--

Implements **decaf::lang::CharSequence** (p. 1168).

**6.784.3.2** `bool decaf::lang::String::isEmpty ( ) const`

#### Returns

true if the length of this **String** (p. 3775) is zero.

**6.784.3.3** `virtual int decaf::lang::String::length ( ) const` [virtual]

#### Returns

the length of the underlying character sequence.

Implements **decaf::lang::CharSequence** (p. 1168).

**6.784.3.4** `virtual CharSequence* decaf::lang::String::subSequence ( int start, int end ) const`  
`throw ( lang::exceptions::IndexOutOfBoundsException )` [virtual]

Returns a new **CharSequence** (p. 1167) that is a subsequence of this sequence.

The subsequence starts with the char value at the specified index and ends with the char value at index end - 1. The length (in chars) of the returned sequence is end - start, so if start == end then an empty sequence is returned.

#### Parameters

<i>start</i>	- the start index, inclusive
<i>end</i>	- the end index, exclusive

#### Returns

a new **CharSequence** (p. 1167)

#### Exceptions

<i>IndexOutOfBoundsException</i>	if start or end > <b>length()</b> (p. 1168) or start or end are negative.
----------------------------------	---

Implements **decaf::lang::CharSequence** (p. 1168).

**6.784.3.5** `virtual std::string decaf::lang::String::toString ( ) const` [virtual]

#### Returns

the string representation of this **CharSequence** (p. 1167)

Implements **decaf::lang::CharSequence** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**String.h**

## 6.785 decaf::util::StringTokenizer Class Reference

```
#include <src/main/decaf/util/StringTokenizer.h>
```

### Public Member Functions

- **StringTokenizer** (const std::string &str, const std::string &delim=" \t\n\r\f", bool returnDelims=false)  
*Constructs a string tokenizer for the specified string.*
- virtual ~**StringTokenizer** ()
- virtual int **countTokens** () const  
*Calculates the number of times that this tokenizer's nextToken method can be called before it generates an exception.*
- virtual bool **hasMoreTokens** () const  
*Tests if there are more tokens available from this tokenizer's string.*
- virtual std::string **nextToken** () throw ( lang::exceptions::NoSuchElementException )  
*Returns the next token from this string tokenizer.*
- virtual std::string **nextToken** (const std::string &delim) throw ( lang::exceptions::NoSuchElementException )  
*Returns the next token in this string tokenizer's string.*
- virtual unsigned int **toArray** (std::vector< std::string > &array)  
*Grab all remaining tokens in the String and return them in the vector that is passed in by reference.*
- virtual void **reset** (const std::string &str="", const std::string &delim="", bool returnDelims=false)  
*Resets the Tokenizer's position in the String to the Beginning calls to countToken and nextToken now start back at the beginning.*

### 6.785.1 Constructor & Destructor Documentation

**6.785.1.1 decaf::util::StringTokenizer::StringTokenizer ( const std::string & str, const std::string & delim = " \t\n\r\f", bool returnDelims = false )**

Constructs a string tokenizer for the specified string.

All characters in the delim argument are the delimiters for separating tokens.

If the returnDelims flag is true, then the delimiter characters are also returned as tokens. Each delimiter is returned as a string of length one. If the flag is false, the delimiter characters are skipped and only serve as separators between tokens.

Note that if `delim` is "", this constructor does not throw an exception. However, trying to invoke other methods on the resulting **StringTokenizer** (p. 3779) may result in an Exception.

#### Parameters

<i>str</i>	- The string to tokenize
<i>delim</i>	- String containing the delimiters
<i>returnDelims</i>	- boolean indicating if the delimiters are returned as tokens

6.785.1.2 `virtual decaf::util::StringTokenizer::~StringTokenizer ( ) [virtual]`

### 6.785.2 Member Function Documentation

6.785.2.1 `virtual int decaf::util::StringTokenizer::countTokens ( ) const [virtual]`

Calculates the number of times that this tokenizer's `nextToken` method can be called before it generates an exception.

The current position is not advanced.

#### Returns

Count of remaining tokens

6.785.2.2 `virtual bool decaf::util::StringTokenizer::hasMoreTokens ( ) const [virtual]`

Tests if there are more tokens available from this tokenizer's string.

#### Returns

true if there are more tokens remaining

6.785.2.3 `virtual std::string decaf::util::StringTokenizer::nextToken ( const std::string & delim ) throw ( lang::exceptions::NoSuchElementException ) [virtual]`

Returns the next token in this string tokenizer's string.

First, the set of characters considered to be delimiters by this **StringTokenizer** (p. 3779) object is changed to be the characters in the string `delim`. Then the next token in the string after the current position is returned. The current position is advanced beyond the recognized token. The new delimiter set remains the default after this call.

#### Parameters

<i>delim</i>	- string containing the new set of delimiters
--------------	---



**Returns**

next string in the token list

**Exceptions**

<i>NoSuchElementException</i>	
-------------------------------	--

**6.785.2.4** virtual std::string decaf::util::StringTokenizer::nextToken ( ) throw ( lang::exceptions::NoSuchElementException ) [virtual]

Returns the next token from this string tokenizer.

**Returns**

string value of next token

**Exceptions**

<i>NoSuchElementException</i>	
-------------------------------	--

**6.785.2.5** virtual void decaf::util::StringTokenizer::reset ( const std::string & str = "", const std::string & delim = " ", bool returnDelims = false ) [virtual]

Resets the Tokenizer's position in the String to the Beginning calls to countToken and nextToken now start back at the beginning.

This allows this object to be reused, the caller need not create a new instance every time a String needs tokenizing. If set the string param will reset the string that this Tokenizer is working on. If set to "" no change is made. If set the delim param will reset the string that this Tokenizer is using to tokenize the string. If set to "", no change is made If set the return Delims will set if this Tokenizer will return delimiters as tokens. Defaults to false.

**Parameters**

<i>str</i>	- New String to tokenize or "", defaults to ""
<i>delim</i>	- New Delimiter String to use or "", defaults to ""
<i>returnDelims</i>	- Should the Tokenizer return delimiters as Tokens, default false

**6.785.2.6** virtual unsigned int decaf::util::StringTokenizer::toArray ( std::vector< std::string > & array ) [virtual]

Grab all remaining tokens in the String and return them in the vector that is passed in by reference.

**Parameters**

<i>array</i>	- vector to place token strings in
--------------	------------------------------------

**Returns**

number of string placed into the vector

The documentation for this class was generated from the following file:

- src/main/decaf/util/**StringTokenizer.h**

**6.786 activemq::commands::SubscriptionInfo Class Reference**

```
#include <src/main/activemq/commands/SubscriptionInfo.h>
```

Inheritance diagram for activemq::commands::SubscriptionInfo:

**Public Member Functions**

- **SubscriptionInfo** ()
- virtual **~SubscriptionInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **SubscriptionInfo \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const

- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &**destination**)
- virtual const std::string & **getSelector** () const
- virtual std::string & **getSelector** ()
- virtual void **setSelector** (const std::string &**selector**)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &**subscriptionName**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getSubscribedDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getSubscribedDestination** ()
- virtual void **setSubscribedDestination** (const **Pointer**< **ActiveMQDestination** > &**subscribedDestination**)

### Static Public Attributes

- static const unsigned char **ID\_SUBSCRIPTIONINFO** = 55

### Protected Attributes

- std::string **clientId**
- **Pointer**< **ActiveMQDestination** > **destination**
- std::string **selector**
- std::string **subscriptionName**
- **Pointer**< **ActiveMQDestination** > **subscribedDestination**

## 6.786.1 Constructor & Destructor Documentation

6.786.1.1 **activemq::commands::SubscriptionInfo::SubscriptionInfo** ( )

6.786.1.2 **virtual activemq::commands::SubscriptionInfo::~~SubscriptionInfo** ( )  
[virtual]

## 6.786.2 Member Function Documentation

6.786.2.1 **virtual SubscriptionInfo\*** **activemq::commands::SubscriptionInfo::cloneDataStructure** ( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

**6.786.2.2** `virtual void activemq::commands::SubscriptionInfo::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Implements `activemq::commands::DataStructure` (p. 1715).

**6.786.2.3** `virtual bool activemq::commands::SubscriptionInfo::equals ( const DataStructure * value ) const [virtual]`

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if `DataStructure`'s are Equal.

Implements `activemq::commands::DataStructure` (p. 1716).

**6.786.2.4** `virtual const std::string& activemq::commands::SubscriptionInfo::getClientId ( ) const [virtual]`

**6.786.2.5** `virtual std::string& activemq::commands::SubscriptionInfo::getClientId ( ) [virtual]`

**6.786.2.6** `virtual unsigned char activemq::commands::SubscriptionInfo::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new `DataStructure` (p. 1713) type copy.

Implements `activemq::commands::DataStructure` (p. 1717).

- 6.786.2.7 **virtual const Pointer<ActiveMQDestination>&**  
**activemq::commands::SubscriptionInfo::getDestination ( ) const** [virtual]
- 6.786.2.8 **virtual Pointer<ActiveMQDestination>&**  
**activemq::commands::SubscriptionInfo::getDestination ( )** [virtual]
- 6.786.2.9 **virtual const std::string& activemq::commands::SubscriptionInfo::getSelector ( )**  
**const** [virtual]
- 6.786.2.10 **virtual std::string& activemq::commands::SubscriptionInfo::getSelector ( )**  
[virtual]
- 6.786.2.11 **virtual std::string& activemq::commands::SubscriptionInfo::getSubscriptionName ( )**  
[virtual]
- 6.786.2.12 **virtual const std::string& activemq::commands::SubscriptionInfo::getSubscriptionName**  
**( ) const** [virtual]
- 6.786.2.13 **virtual const Pointer<ActiveMQDestination>&**  
**activemq::commands::SubscriptionInfo::getSubscribedDestination ( ) const**  
[virtual]
- 6.786.2.14 **virtual Pointer<ActiveMQDestination>&**  
**activemq::commands::SubscriptionInfo::getSubscribedDestination ( )**  
[virtual]
- 6.786.2.15 **virtual void activemq::commands::SubscriptionInfo::setClientId ( const std::string &**  
**clientId )** [virtual]
- 6.786.2.16 **virtual void activemq::commands::SubscriptionInfo::setDestination ( const**  
**Pointer< ActiveMQDestination > & destination )** [virtual]
- 6.786.2.17 **virtual void activemq::commands::SubscriptionInfo::setSelector ( const std::string &**  
**selector )** [virtual]
- 6.786.2.18 **virtual void activemq::commands::SubscriptionInfo::setSubscriptionName ( const**  
**std::string & subscriptionName )** [virtual]
- 6.786.2.19 **virtual void activemq::commands::SubscriptionInfo::setSubscribedDestination**  
**( const Pointer< ActiveMQDestination > & subscribedDestination )**  
[virtual]
- 6.786.2.20 **virtual std::string activemq::commands::SubscriptionInfo::toString ( ) const**  
[virtual]

Returns a string containing the information for this **DataSet** (p. 1713) such as its type and value of its elements.

**Returns**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 841).

**6.786.3 Field Documentation**

**6.786.3.1** `std::string activemq::commands::SubscriptionInfo::clientId`  
[protected]

**6.786.3.2** `Pointer<ActiveMQDestination> activemq::commands::SubscriptionInfo::destination`  
[protected]

**6.786.3.3** `const unsigned char activemq::commands::SubscriptionInfo::ID_-SUBSCRIPTIONINFO = 55` [static]

**6.786.3.4** `std::string activemq::commands::SubscriptionInfo::selector`  
[protected]

**6.786.3.5** `std::string activemq::commands::SubscriptionInfo::subscriptionName`  
[protected]

**6.786.3.6** `Pointer<ActiveMQDestination> activemq::commands::SubscriptionInfo::subscribedDestination`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/SubscriptionInfo.h`

**6.787 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller Class Reference**

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3786).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInf
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller`:

**Public Member Functions**

- **SubscriptionInfoMarshaller ()**

6.787

activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller

Class Reference

3791

- virtual ~SubscriptionInfoMarshaller ()

- virtual commands::DataStructure \* createObject () const

*Creates a new instance of this marshalable type.*

- virtual unsigned char getDataStructureType () const

*Get the Data Structure Type that identifies this Marshaler.*

- virtual void tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual int tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.787.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3786).  
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.787.2 Constructor & Destructor Documentation

6.787.2.1 `activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller ( ) [inline]`

6.787.2.2 `virtual activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller ( ) [inline, virtual]`

## 6.787.3 Member Function Documentation

6.787.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.787.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.787.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



6.787

**activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller**

**Class Reference**

**3793**

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.787.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.787.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```

6.787.3.6 virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```

6.787.3.7 virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**SubscriptionInfoMarshaller.h**

6.788

activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller

Class Reference

6.788 — activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller <sup>3795</sup>

## Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3791).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller:

## Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.788.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3791).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.788.2 Constructor & Destructor Documentation

6.788.2.1 **activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller**  
 ( ) [inline]

6.788.2.2 **virtual activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller**  
 ( ) [inline, virtual]

### 6.788.3 Member Function Documentation

6.788.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::createObject (**  
**) const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.788.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::getDataStructureType**  
**( ) const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.788.3.3 **virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure**  
**\* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw (**  
**decaf::io::IOException )** [virtual]

Write a object instance to data output stream.

#### Parameters

6.788

activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller

Class Reference

3797

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.788.3.4** virtual void **activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.788.3.5** virtual int **activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::tightMarshal1** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.788.3.6  virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.788.3.7  virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

6.789

activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller

Class Reference

3799

- src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h

## 6.789 activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3795).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller:

### Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.789.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3795).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.789.2 Constructor & Destructor Documentation

6.789.2.1 **activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller**  
 ( ) [inline]

6.789.2.2 **virtual activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller**  
 ( ) [inline, virtual]

### 6.789.3 Member Function Documentation

6.789.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::createObject (**  
**) const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.789.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::getDataStructureType**  
**( ) const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.789.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure**  
**\* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw (**  
**decaf::io::IOException )** [virtual]

Write a object instance to data output stream.



6.789

**activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller**

**Class Reference**

**3801**

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.789.3.4** virtual void **activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.789.3.5** virtual int **activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::tightMarshal1** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.789.3.6  virtual void activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.789.3.7  virtual void activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

6.790

activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller

Class Reference

3803

- src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h

## 6.790 activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3799).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller:

### Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.790.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3799).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.790.2 Constructor & Destructor Documentation

6.790.2.1 **activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller**  
 ( ) [inline]

6.790.2.2 **virtual activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller**  
 ( ) [inline, virtual]

### 6.790.3 Member Function Documentation

6.790.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::createObject (**  
**) const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.790.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::getDataStructureType**  
**( ) const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.790.3.3 **virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure**  
**\* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw (**  
**decaf::io::IOException )** [virtual]

Write a object instance to data output stream.

6.790

**activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller**

**Class Reference**

**3805**

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.790.3.4** virtual void **activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.790.3.5** virtual int **activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::tightMarshal1** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.790.3.6  virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.790.3.7  virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

6.791

activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller

Class Reference

3807

- src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h

## 6.791 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3803).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/SubscriptionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller:

### Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.791.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3803).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.791.2 Constructor & Destructor Documentation

6.791.2.1 **activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller**  
 ( ) [inline]

6.791.2.2 **virtual activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller**  
 ( ) [inline, virtual]

### 6.791.3 Member Function Documentation

6.791.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::createObject (**  
**) const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.791.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::getDataStructureType**  
**( ) const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.791.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure**  
**\* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw (**  
**decaf::io::IOException )** [virtual]

Write a object instance to data output stream.



6.791

activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller

Class Reference

3809

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.791.3.4** virtual void activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::looseUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.791.3.5** virtual int activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::tightMarshal1 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.791.3.6  virtual void activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.791.3.7  virtual void activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

6.792

activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller

Class Reference

3811

- src/main/activemq/wireformat/openwire/marshal/v6/SubscriptionInfoMarshaller.h

## 6.792 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3807).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller:

### Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.792.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3807).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.792.2 Constructor & Destructor Documentation

6.792.2.1 **activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller**  
 ( ) [inline]

6.792.2.2 **virtual activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller**  
 ( ) [inline, virtual]

### 6.792.3 Member Function Documentation

6.792.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::createObject (**  
**) const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.792.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::getDataStructureType**  
**( ) const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.792.3.3 **virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure**  
**\* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw (**  
**decaf::io::IOException )** [virtual]

Write a object instance to data output stream.

6.792

**activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller**

**Class Reference**

**3813**

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.792.3.4** virtual void **activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.792.3.5** virtual int **activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::tightMarshal1** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.792.3.6  virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.792.3.7  virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h

## 6.793 decaf::util::concurrent::Synchronizable Class Reference

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

```
#include <src/main/decaf/util/concurrent/Synchronizable.h>
```

Inheritance diagram for decaf::util::concurrent::Synchronizable:

### Public Member Functions

- virtual **~Synchronizable** ()
- virtual void **lock** ()=0 throw ( decaf::lang::exceptions::RuntimeException )  
*Locks the object.*
- virtual bool **tryLock** ()=0 throw ( decaf::lang::exceptions::RuntimeException )  
*Attempts to **Lock** (p. 2450) the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** ()=0 throw ( decaf::lang::exceptions::RuntimeException )  
*Unlocks the object.*
- virtual void **wait** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs)=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs, int nanos)=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **notify** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals a waiter on this object that it can now wake up and continue.*
- virtual void **notifyAll** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals the waiters on this object that it can now wake up and continue.*

### 6.793.1 Detailed Description

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

Since

1.0

### 6.793.2 Constructor & Destructor Documentation

**6.793.2.1** `virtual decaf::util::concurrent::Synchronizable::~~Synchronizable ( ) [inline, virtual]`

### 6.793.3 Member Function Documentation

**6.793.3.1** `virtual void decaf::util::concurrent::Synchronizable::lock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [pure virtual]`

Locks the object.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implemented in `activemq::core::MessageDispatchChannel` (p. 2687), `decaf::internal::util::concurrent::Synchronizable` (p. 3823), `decaf::io::InputStream` (p. 2109), `decaf::io::OutputStream` (p. 2995), `decaf::util::AbstractCollection< E >` (p. 167), `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1274), `decaf::util::concurrent::Mutex` (p. 2868), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3717), `decaf::util::StlQueue< T >` (p. 3727), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 167), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 167), `decaf::util::AbstractCollection< Resource * >` (p. 167), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 167), `decaf::util::AbstractCollection< CompositeTask * >` (p. 167), `decaf::util::AbstractCollection< URI >` (p. 167), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 167), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 167), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 167), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 167), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 167), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 167), `decaf::util::AbstractCollection< cms::Destination * >` (p. 167), `decaf::util::AbstractCollection< cms::Session * >` (p. 167), `decaf::util::AbstractCollection< cms::Connection * >` (p. 167), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1274), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1274), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId`



>, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1274), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1274), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1274), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1274), decaf::util::StlMap< cms::Session \*, SessionResolver \* > (p. 3717), decaf::util::StlMap< std::string, WireFormatFactory \* > (p. 3717), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3717), decaf::util::StlMap< std::string, cms::Queue \* > (p. 3717), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR > (p. 3717), decaf::util::StlMap< std::string, CachedConsumer \* > (p. 3717), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR > (p. 3717), decaf::util::StlMap< std::string, TransportFactory \* > (p. 3717), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3717), decaf::util::StlMap< int, Pointer< Command > > (p. 3717), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR > (p. 3717), decaf::util::StlMap< std::string, CachedProducer \* > (p. 3717), decaf::util::StlMap< std::string, cms::Topic \* > (p. 3717), decaf::util::StlQueue< Pointer< Transport > > (p. 3727), decaf::util::StlQueue< Pointer< MessageDispatch > > (p. 3727), decaf::util::StlQueue< Task > (p. 3727), decaf::util::StlQueue< Pointer< Command > > (p. 3727), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > > (p. 3727).

```
6.793.3.2 virtual void decaf::util::concurrent::Synchronizable::notify ( )
            throw ( decaf::lang::exceptions::RuntimeException,
                    decaf::lang::exceptions::IllegalMonitorStateException ) [pure
            virtual]
```

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3811) Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2687), **decaf::internal::util::concurrent::Synchronizable** (p. 3823), **decaf::io::InputStream** (p. 2110), **decaf::io::OutputStream** (p. 2995), **decaf::util::AbstractCollection< E >** (p. 167), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1274), **decaf::util::concurrent::Mutex** (p. 2868), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3717), **decaf::util::StlQueue< T >** (p. 3727), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 167), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 167), **decaf::util::AbstractCollection< Resource \* >** (p. 167), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 167), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 167), **decaf::util::AbstractCollection< URI >** (p. 167), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 167), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 167), **decaf::util::AbstractCollection<**

**PrimitiveValueNode** > > (p. 167), **decaf::util::AbstractCollection**< **Pointer**< **Command** > > (p. 167), **decaf::util::AbstractCollection**< **Pointer**< **BackupTransport** > > (p. 167), **decaf::util::AbstractCollection**< **cms::MessageProducer** \* > (p. 167), **decaf::util::AbstractCollection**< **cms::Destination** \* > (p. 167), **decaf::util::AbstractCollection**< **cms::Session** \* > (p. 167), **decaf::util::AbstractCollection**< **cms::Connection** \* > (p. 167), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **MessageId** >, **Pointer**< **Message** >, **MessageId::COMPARATOR** > (p. 1274), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ConnectionId** >, **Pointer**< **ConnectionState** >, **ConnectionId::COMPARATOR** > (p. 1274), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ConsumerId** >, **Pointer**< **ConsumerState** >, **ConsumerId::COMPARATOR** > (p. 1274), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **SessionId** >, **Pointer**< **SessionState** >, **SessionId::COMPARATOR** > (p. 1274), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **LocalTransactionId** >, **Pointer**< **TransactionState** >, **LocalTransactionId::COMPARATOR** > (p. 1274), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ProducerId** >, **Pointer**< **ProducerState** >, **ProducerId::COMPARATOR** > (p. 1274), **decaf::util::StlMap**< **cms::Session** \*, **SessionResolver** \* > (p. 3717), **decaf::util::StlMap**< **std::string**, **WireFormatFactory** \* > (p. 3717), **decaf::util::StlMap**< **std::string**, **PrimitiveValueNode** > (p. 3717), **decaf::util::StlMap**< **std::string**, **cms::Queue** \* > (p. 3717), **decaf::util::StlMap**< **Pointer**< **commands::ProducerId** >, **ActiveMQProducer** \*, **commands::ProducerId::COMPARATOR** > (p. 3717), **decaf::util::StlMap**< **std::string**, **CachedConsumer** \* > (p. 3717), **decaf::util::StlMap**< **Pointer**< **commands::ConsumerId** >, **ActiveMQConsumer** \*, **commands::ConsumerId::COMPARATOR** > (p. 3717), **decaf::util::StlMap**< **std::string**, **TransportFactory** \* > (p. 3717), **decaf::util::StlMap**< **Pointer**< **ConsumerId** >, **Pointer**< **ConsumerInfo** >, **ConsumerId::COMPARATOR** > (p. 3717), **decaf::util::StlMap**< **int**, **Pointer**< **Command** > > (p. 3717), **decaf::util::StlMap**< **Pointer**< **commands::ConsumerId** >, **Dispatcher** \*, **commands::ConsumerId::COMPARATOR** > (p. 3717), **decaf::util::StlMap**< **std::string**, **CachedProducer** \* > (p. 3717), **decaf::util::StlMap**< **std::string**, **cms::Topic** \* > (p. 3717), **decaf::util::StlQueue**< **Pointer**< **Transport** > > (p. 3727), **decaf::util::StlQueue**< **Pointer**< **MessageDispatch** > > (p. 3727), **decaf::util::StlQueue**< **Task** > (p. 3727), **decaf::util::StlQueue**< **Pointer**< **Command** > > (p. 3727), and **decaf::util::StlQueue**< **decaf::lang::Pointer**< **commands::MessageDispatch** > > (p. 3727).

```

6.793.3.3  virtual void decaf::util::concurrent::Synchronizable::notifyAll ( )
            throw ( decaf::lang::exceptions::RuntimeException,
                    decaf::lang::exceptions::IllegalMonitorStateException ) [pure
            virtual]
  
```

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3811) Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2687), **decaf::internal::util::concurrent::Synchronizable** (p. 3824), **decaf::io::InputStream** (p. 2110), **decaf::io::OutputStream** (p. 2995),

decaf::util::AbstractCollection< E > (p. 167), decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1275), decaf::util::concurrent::Mutex (p. 2868), decaf::util::StlMap< K, V, COMPARATOR > (p. 3717), decaf::util::StlQueue< T > (p. 3727), decaf::util::AbstractCollection< transport::TransportListener \* > (p. 167), decaf::util::AbstractCollection< Pointer< Synchronization > > (p. 167), decaf::util::AbstractCollection< Resource \* > (p. 167), decaf::util::AbstractCollection< cms::MessageConsumer \* > (p. 167), decaf::util::AbstractCollection< CompositeTask \* > (p. 167), decaf::util::AbstractCollection< URI > (p. 167), decaf::util::AbstractCollection< ActiveMQSession \* > (p. 167), decaf::util::AbstractCollection< Pointer< DestinationInfo > > (p. 167), decaf::util::AbstractCollection< PrimitiveValueNode > (p. 167), decaf::util::AbstractCollection< Pointer< Command > > (p. 167), decaf::util::AbstractCollection< Pointer< BackupTransport > > (p. 167), decaf::util::AbstractCollection< cms::MessageProducer \* > (p. 167), decaf::util::AbstractCollection< cms::Destination \* > (p. 167), decaf::util::AbstractCollection< cms::Session \* > (p. 167), decaf::util::AbstractCollection< cms::Connection \* > (p. 167), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1275), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1275), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1275), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1275), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1275), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1275), decaf::util::StlMap< cms::Session \*, SessionResolver \* > (p. 3717), decaf::util::StlMap< std::string, WireFormatFactory \* > (p. 3717), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3717), decaf::util::StlMap< std::string, cms::Queue \* > (p. 3717), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR > (p. 3717), decaf::util::StlMap< std::string, CachedConsumer \* > (p. 3717), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR > (p. 3717), decaf::util::StlMap< std::string, TransportFactory \* > (p. 3717), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3717), decaf::util::StlMap< int, Pointer< Command > > (p. 3717), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR > (p. 3717), decaf::util::StlMap< std::string, CachedProducer \* > (p. 3717), decaf::util::StlMap< std::string, cms::Topic \* > (p. 3717), decaf::util::StlQueue< Pointer< Transport > > (p. 3727), decaf::util::StlQueue< Pointer< MessageDispatch > > (p. 3727), decaf::util::StlQueue< Task > (p. 3727), decaf::util::StlQueue< Pointer< Command > > (p. 3727), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > > (p. 3727).

**6.793.3.4** virtual bool decaf::util::concurrent::Synchronizable::tryLock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [pure virtual]

Attempts to **Lock** (p. 2450) the object, if the lock is already held by another thread than this method returns false.

### Returns

true if the lock was acquired, false if it is already held by another thread.

### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implemented in `activemq::core::MessageDispatchChannel` (p. 2688), `decaf::internal::util::concurrent::Sync` (p. 3824), `decaf::io::InputStream` (p. 2114), `decaf::io::OutputStream` (p. 2996), `decaf::util::AbstractCollection< E >` (p. 171), `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1280), `decaf::util::concurrent::Mutex` (p. 2869), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3719), `decaf::util::StlQueue< T >` (p. 3729), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 171), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 171), `decaf::util::AbstractCollection< Resource * >` (p. 171), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 171), `decaf::util::AbstractCollection< CompositeTask * >` (p. 171), `decaf::util::AbstractCollection< URI >` (p. 171), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 171), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 171), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 171), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 171), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 171), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 171), `decaf::util::AbstractCollection< cms::Destination * >` (p. 171), `decaf::util::AbstractCollection< cms::Session * >` (p. 171), `decaf::util::AbstractCollection< cms::Connection * >` (p. 171), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1280), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1280), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1280), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1280), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1280), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1280), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3719), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3719), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3719), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3719), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3719), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3719), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3719), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3719), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3719), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3719), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3719), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3719), `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3719), `decaf::util::StlQueue< Pointer< Transport > >` (p. 3729), `decaf::util::StlQueue< Pointer< MessageDispatch > >` (p. 3729), `decaf::util::StlQueue< Task >` (p. 3729), `decaf::util::StlQueue< Pointer< Command > >` (p. 3729), and `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >` (p. 3729).

**6.793.3.5** virtual void decaf::util::concurrent::Synchronizable::unlock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [pure virtual]

Unlocks the object.

### Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implemented in **activemq::core::MessageDispatchChannel** (p. 2689), **decaf::internal::util::concurrent::Synchronizable** (p. 3824), **decaf::io::InputStream** (p. 2115), **decaf::io::OutputStream** (p. 2996), **decaf::util::AbstractCollection< E >** (p. 171), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1280), **decaf::util::concurrent::Mutex** (p. 2869), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3720), **decaf::util::StlQueue< T >** (p. 3729), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 171), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 171), **decaf::util::AbstractCollection< Resource \* >** (p. 171), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 171), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 171), **decaf::util::AbstractCollection< URI >** (p. 171), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 171), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 171), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 171), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 171), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 171), **decaf::util::AbstractCollection< cms::MessageProducer \* >** (p. 171), **decaf::util::AbstractCollection< cms::Destination \* >** (p. 171), **decaf::util::AbstractCollection< cms::Session \* >** (p. 171), **decaf::util::AbstractCollection< cms::Connection \* >** (p. 171), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1280), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1280), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1280), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1280), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 1280), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1280), **decaf::util::StlMap< cms::Session \*, SessionResolver \* >** (p. 3720), **decaf::util::StlMap< std::string, WireFormatFactory \* >** (p. 3720), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 3720), **decaf::util::StlMap< std::string, cms::Queue \* >** (p. 3720), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR >** (p. 3720), **decaf::util::StlMap< std::string, CachedConsumer \* >** (p. 3720), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR >** (p. 3720), **decaf::util::StlMap< std::string, TransportFactory \* >** (p. 3720), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 3720), **decaf::util::StlMap< int, Pointer< Command > >** (p. 3720), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR >** (p. 3720), **decaf::util::StlMap< std::string, CachedProducer \* >** (p. 3720), **decaf::util::StlMap< std::string, cms::Topic \* >** (p. 3720), **decaf::util::StlQueue< Pointer< Transport > >** (p. 3729), **decaf::util::StlQueue< Pointer< MessageDispatch > >** (p. 3729), **decaf::util::StlQueue< Task >** (p. 3729), **decaf::util::StlQueue<**

**Pointer< Command > >** (p. 3729), and **decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >** (p. 3729).

6.793.3.6 **virtual void decaf::util::concurrent::Synchronizable::wait ( )**  
**throw ( decaf::lang::exceptions::RuntimeException,**  
**decaf::lang::exceptions::IllegalMonitorStateException,**  
**decaf::lang::exceptions::InterruptedException )** [pure virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3811) Object.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2690), **decaf::internal::util::concurrent::Synchronizable** (p. 3825), **decaf::io::InputStream** (p. 2115), **decaf::io::OutputStream** (p. 2996), **decaf::util::AbstractCollection< E >** (p. 171), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1281), **decaf::util::concurrent::Mutex** (p. 2870), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3720), **decaf::util::StlQueue< T >** (p. 3729), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 171), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 171), **decaf::util::AbstractCollection< Resource \* >** (p. 171), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 171), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 171), **decaf::util::AbstractCollection< URI >** (p. 171), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 171), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 171), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 171), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 171), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 171), **decaf::util::AbstractCollection< cms::MessageProducer \* >** (p. 171), **decaf::util::AbstractCollection< cms::Destination \* >** (p. 171), **decaf::util::AbstractCollection< cms::Session \* >** (p. 171), **decaf::util::AbstractCollection< cms::Connection \* >** (p. 171), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1281), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1281), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1281), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1281), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 1281), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1281), **decaf::util::StlMap< cms::Session \*, SessionResolver \* >** (p. 3720), **decaf::util::StlMap< std::string, WireFormatFactory \* >** (p. 3720), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 3720), **decaf::util::StlMap< std::string, cms::Queue \* >** (p. 3720), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR >** (p. 3720), **decaf::util::StlMap< std::string,**

**CachedConsumer \*** > (p. 3720), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR >** (p. 3720), **decaf::util::StlMap< std::string, TransportFactory \* >** (p. 3720), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 3720), **decaf::util::StlMap< int, Pointer< Command > >** (p. 3720), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR >** (p. 3720), **decaf::util::StlMap< std::string, CachedProducer \* >** (p. 3720), **decaf::util::StlMap< std::string, cms::Topic \* >** (p. 3720), **decaf::util::StlQueue< Pointer< Transport > >** (p. 3729), **decaf::util::StlQueue< Pointer< MessageDispatch > >** (p. 3729), **decaf::util::StlQueue< Task >** (p. 3729), **decaf::util::StlQueue< Pointer< Command > >** (p. 3729), and **decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >** (p. 3729).

**6.793.3.7** virtual void decaf::util::concurrent::Synchronizable::wait ( long long *millisecs* ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [pure virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3811) Object.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2689), **decaf::internal::util::concurrent::Synchronizable** (p. 3825), **decaf::io::InputStream** (p. 2115), **decaf::io::OutputStream** (p. 2997), **decaf::util::AbstractCollection< E >** (p. 172), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1281), **decaf::util::concurrent::Mutex** (p. 2870), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3721), **decaf::util::StlQueue< T >** (p. 3730), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 172), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 172), **decaf::util::AbstractCollection< Resource \* >** (p. 172), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 172), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 172), **decaf::util::AbstractCollection< URI >** (p. 172), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 172), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 172), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 172), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 172), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 172), **decaf::util::AbstractCollection< cms::MessageProducer \* >** (p. 172), **decaf::util::AbstractCollection< cms::Destination \* >** (p. 172), **decaf::util::AbstractCollection< cms::Session \* >** (p. 172), **decaf::util::AbstractCollection< cms::Connection \* >**

> (p. 172), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1281), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1281), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1281), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1281), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1281), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1281), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3721), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3721), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3721), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3721), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3721), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3721), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3721), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3721), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3721), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3721), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3721), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3721), `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3721), `decaf::util::StlQueue< Pointer< Transport > >` (p. 3730), `decaf::util::StlQueue< Pointer< MessageDispatch > >` (p. 3730), `decaf::util::StlQueue< Task >` (p. 3730), `decaf::util::StlQueue< Pointer< Command > >` (p. 3730), and `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >` (p. 3730).

**6.793.3.8** `virtual void decaf::util::concurrent::Synchronizable::wait ( long long milliseconds, int nanos ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )` [pure virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

#### Parameters

<i>milliseconds</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999



## Exceptions

<i>IllegalArgumentException</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3811) Object.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2689), **decaf::internal::util::concurrent::Synchronizable** (p. 3825), **decaf::io::InputStream** (p. 2116), **decaf::io::OutputStream** (p. 2997), **decaf::util::AbstractCollection< E >** (p. 172), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1282), **decaf::util::concurrent::Mutex** (p. 2871), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3721), **decaf::util::StlQueue< T >** (p. 3730), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 172), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 172), **decaf::util::AbstractCollection< Resource \* >** (p. 172), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 172), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 172), **decaf::util::AbstractCollection< URI >** (p. 172), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 172), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 172), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 172), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 172), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 172), **decaf::util::AbstractCollection< cms::MessageProducer \* >** (p. 172), **decaf::util::AbstractCollection< cms::Destination \* >** (p. 172), **decaf::util::AbstractCollection< cms::Session \* >** (p. 172), **decaf::util::AbstractCollection< cms::Connection \* >** (p. 172), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1282), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1282), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1282), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1282), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 1282), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1282), **decaf::util::StlMap< cms::Session \*, SessionResolver \* >** (p. 3721), **decaf::util::StlMap< std::string, WireFormatFactory \* >** (p. 3721), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 3721), **decaf::util::StlMap< std::string, cms::Queue \* >** (p. 3721), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR >** (p. 3721), **decaf::util::StlMap< std::string, CachedConsumer \* >** (p. 3721), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR >** (p. 3721), **decaf::util::StlMap< std::string, TransportFactory \* >** (p. 3721), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 3721), **decaf::util::StlMap< int, Pointer< Command > >** (p. 3721), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR >** (p. 3721), **decaf::util::StlMap< std::string, CachedProducer \* >** (p. 3721), **decaf::util::StlMap< std::string, cms::Topic \* >** (p. 3721), **decaf::util::StlQueue< Pointer< Transport > >** (p. 3730), **decaf::util::StlQueue< Pointer< MessageDispatch > >** (p. 3730), **decaf::util::StlQueue< Task >** (p. 3730), **decaf::util::StlQueue<**

**Pointer**< **Command** > > (p. 3730), and **decaf::util::StlQueue**< **decaf::lang::Pointer**< **commands::MessageDispatch** > > (p. 3730).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Synchronizable.h`

## 6.794 **decaf::internal::util::concurrent::SynchronizableImpl** Class Reference

A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

```
#include <src/main/decaf/internal/util/concurrent/SynchronizableImpl.h>
```

Inheritance diagram for **decaf::internal::util::concurrent::SynchronizableImpl**:

### Public Member Functions

- **SynchronizableImpl** ()
- virtual **~SynchronizableImpl** ()
- virtual void **lock** () throw ( **decaf::lang::exceptions::RuntimeException** )  
*Locks the object.*
- virtual bool **tryLock** () throw ( **decaf::lang::exceptions::RuntimeException** )  
*Attempts to Lock the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** () throw ( **decaf::lang::exceptions::RuntimeException** )  
*Unlocks the object.*
- virtual void **wait** () throw ( **decaf::lang::exceptions::RuntimeException**, **decaf::lang::exceptions::IllegalMonitorStateException**, **decaf::lang::exceptions::InterruptedException** )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs) throw ( **decaf::lang::exceptions::RuntimeException**, **decaf::lang::exceptions::IllegalMonitorStateException**, **decaf::lang::exceptions::InterruptedException** )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs, int nanos) throw ( **decaf::lang::exceptions::RuntimeException**, **decaf::lang::exceptions::IllegalArgumentOutOfRangeException**, **decaf::lang::exceptions::IllegalMonitorStateException**, **decaf::lang::exceptions::InterruptedException** )  
*Waits on a signal from this object, which is generated by a call to Notify.*

## 6.794 decaf::internal::util::concurrent::SynchronizableImpl Class Reference 3827

- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals a waiter on this object that it can now wake up and continue.*

- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals the waiters on this object that it can now wake up and continue.*

### 6.794.1 Detailed Description

A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

Since

1.0

### 6.794.2 Constructor & Destructor Documentation

6.794.2.1 decaf::internal::util::concurrent::SynchronizableImpl::SynchronizableImpl ( )

6.794.2.2 virtual decaf::internal::util::concurrent::SynchronizableImpl::~SynchronizableImpl ( ) [virtual]

### 6.794.3 Member Function Documentation

6.794.3.1 virtual void decaf::internal::util::concurrent::SynchronizableImpl::lock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [virtual]

Locks the object.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements decaf::util::concurrent::Synchronizable (p. 3812).

6.794.3.2 virtual void decaf::internal::util::concurrent::SynchronizableImpl::notify ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException ) [virtual]

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

**Exceptions**

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3813).

**6.794.3.3** `virtual void decaf::internal::util::concurrent::SynchronizableImpl::notifyAll ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException ) [virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

**Exceptions**

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3814).

**6.794.3.4** `virtual bool decaf::internal::util::concurrent::SynchronizableImpl::tryLock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

**Returns**

true if the lock was acquired, false if it is already held by another thread.

**Exceptions**

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3815).

**6.794.3.5** `virtual void decaf::internal::util::concurrent::SynchronizableImpl::unlock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [virtual]`

Unlocks the object.

**Exceptions**

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

## 6.794 decaf::internal::util::concurrent::SynchronizableImpl Class Reference 3829

Implements **decaf::util::concurrent::Synchronizable** (p. 3816).

**6.794.3.6** virtual void decaf::internal::util::concurrent::SynchronizableImpl::wait ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3818).

**6.794.3.7** virtual void decaf::internal::util::concurrent::SynchronizableImpl::wait ( long long *millisecs* ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3819).

6.794.3.8 `virtual void decaf::internal::util::concurrent::SynchronizableImpl::wait ( long long millisecs, int nanos ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

#### Exceptions

<i>IllegalArgumentException</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 3820).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/concurrent/SynchronizableImpl.h`

## 6.795 activemq::core::Synchronization Class Reference

Transacted Object **Synchronization** (p. 3826), used to sync the events of a Transaction with the items in the Transaction.

```
#include <src/main/activemq/core/Synchronization.h>
```

#### Public Member Functions

- `virtual ~Synchronization ()`
- `virtual void beforeEnd ()=0 throw ( exceptions::ActiveMQException )`
- `virtual void afterCommit ()=0 throw ( exceptions::ActiveMQException )`
- `virtual void afterRollback ()=0 throw ( exceptions::ActiveMQException )`

### 6.795.1 Detailed Description

Transacted Object **Synchronization** (p. 3826), used to sync the events of a Transaction with the items in the Transaction.

### 6.795.2 Constructor & Destructor Documentation

6.795.2.1 `virtual activemq::core::Synchronization::~Synchronization ( ) [inline, virtual]`

### 6.795.3 Member Function Documentation

6.795.3.1 `virtual void activemq::core::Synchronization::afterCommit ( ) throw ( exceptions::ActiveMQException ) [pure virtual]`

6.795.3.2 `virtual void activemq::core::Synchronization::afterRollback ( ) throw ( exceptions::ActiveMQException ) [pure virtual]`

6.795.3.3 `virtual void activemq::core::Synchronization::beforeEnd ( ) throw ( exceptions::ActiveMQException ) [pure virtual]`

The documentation for this class was generated from the following file:

- `src/main/activemq/core/Synchronization.h`

## 6.796 decaf::util::concurrent::SynchronousQueue< E > Class Template Reference

A **blocking queue** (p. 848) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.

```
#include <src/main/decaf/util/concurrent/SynchronousQueue.h>
```

Inheritance diagram for `decaf::util::concurrent::SynchronousQueue< E >`:

### Data Structures

- class **EmptyIterator**

### Public Member Functions

- **SynchronousQueue** ( )
- virtual **~SynchronousQueue** ( )

- virtual void **put** (const E &value) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Adds the specified element to this queue, waiting if necessary for another thread to receive it.*
- virtual bool **offer** (const E &e, long timeout, const **TimeUnit** &unit) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Inserts the specified element into this queue, waiting if necessary up to the specified wait time for another thread to receive it.*
- virtual bool **offer** (const E &value) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Inserts the specified element into this queue, if another thread is waiting to receive it.*
- virtual E **take** () throw ( decaf::lang::exceptions::InterruptedException )  
*Retrieves and removes the head of this queue, waiting if necessary for another thread to insert it.*
- virtual bool **poll** (E &result, long long timeout, const **TimeUnit** &unit) throw ( decaf::lang::exceptions::InterruptedException )  
*Retrieves and removes the head of this queue, waiting if necessary up to the specified wait time, for another thread to insert it.*
- virtual bool **poll** (E &result)  
*Retrieves and removes the head of this queue, if another thread is currently making an element available.*
- virtual bool **equals** (const **Collection**< E > &value) const  
*Answers true if this **Collection** (p.1216) and the one given are the same size and if each element contained in the **Collection** (p.1216) given is equal to an element contained in this collection.*
- virtual **decaf::util::Iterator**< E > \* **iterator** ()
- virtual **decaf::util::Iterator**< E > \* **iterator** () const
- virtual bool **isEmpty** () const  
*Returns true if this collection contains no elements.*
- virtual std::size\_t **size** () const  
*Returns the number of elements in this collection.*
- virtual int **remainingCapacity** () const  
*Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or Integer::MAX\_VALUE if there is no intrinsic limit.*



- virtual void **clear** () throw ( lang::exceptions::UnsupportedOperationException )  
*Removes all elements of the queue.*
- virtual bool **contains** (const E &value DECAF\_UNUSED) const throw ( lang::Exception )
- virtual bool **containsAll** (const **Collection**< E > &collection) const throw ( lang::Exception )  
*Returns true if this collection contains all of the elements in the specified collection.*
- virtual bool **remove** (const E &value DECAF\_UNUSED) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )
- virtual bool **removeAll** (const **Collection**< E > &collection DECAF\_UNUSED) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )
- virtual bool **retainAll** (const **Collection**< E > &collection DECAF\_UNUSED) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )
- virtual bool **peek** (E &result DECAF\_UNUSED) const
- virtual std::vector< E > **toArray** () const  
*Answers an STL vector containing copies of all elements contained in this **Collection** (p. 1216).*
- virtual std::size\_t **drainTo** (**Collection**< E > &c) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException )  
*Removes all available elements from this queue and adds them to the given collection.*
- virtual std::size\_t **drainTo** (**Collection**< E > &c, std::size\_t maxElements) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException )  
*Removes at most the given number of available elements from this queue and adds them to the given collection.*

### 6.796.1 Detailed Description

template<typename E> class decaf::util::concurrent::SynchronousQueue< E >

A **blocking queue** (p. 848) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa. A synchronous queue does not have any internal capacity, not even a capacity of one. You cannot **peek** at a synchronous queue because an element is only present when you try to remove it; you cannot insert an element (using any method) unless another thread is trying to remove it; you cannot iterate as there is nothing to iterate. The *head* of the queue is the element that the first queued inserting thread is trying to add to the queue; if there is no such queued thread then no element is available for removal and **poll()** (p. 3834) will return **null**. For purposes of other **Collection** (p. 1216) methods (for example

contains), a **SynchronousQueue** (p. 3827) acts as an empty collection. This queue does not permit `null` elements.

Synchronous queues are similar to rendezvous channels used in CSP and Ada. They are well suited for handoff designs, in which an object running in one thread must sync up with an object running in another thread in order to hand it some information, event, or task.

This class supports an optional fairness policy for ordering waiting producer and consumer threads. By default, this ordering is not guaranteed. However, a queue constructed with fairness set to `true` grants threads access in FIFO order.

This class and its iterator implement all of the *optional* methods of the **Collection** (p. 1216) and **Iterator** (p. 2222) interfaces.

Since

1.0

## 6.796.2 Constructor & Destructor Documentation

6.796.2.1 `template<typename E> decaf::util::concurrent::SynchronousQueue< E>::SynchronousQueue( ) [inline]`

6.796.2.2 `template<typename E> virtual decaf::util::concurrent::SynchronousQueue< E>::~SynchronousQueue( ) [inline, virtual]`

## 6.796.3 Member Function Documentation

6.796.3.1 `template<typename E> virtual void decaf::util::concurrent::SynchronousQueue< E>::clear( ) throw ( lang::exceptions::UnsupportedOperationException ) [inline, virtual]`

Removes all elements of the queue.

This implementation repeatedly invokes `poll` until it returns the empty marker.

Reimplemented from `decaf::util::AbstractQueue< E>` (p. 178).

6.796.3.2 `template<typename E> virtual bool decaf::util::concurrent::SynchronousQueue< E>::contains( const E &value DECAF_UNUSED ) const throw ( lang::Exception ) [inline, virtual]`

6.796.3.3 `template<typename E> virtual bool decaf::util::concurrent::SynchronousQueue< E>::containsAll( const Collection< E> &collection ) const throw ( lang::Exception ) [inline, virtual]`

Returns true if this collection contains all of the elements in the specified collection.

This implementation iterates over the specified collection, checking each element returned by the iterator in turn to see if it's contained in this collection. If all elements are so contained true is returned, otherwise false.

#### Parameters

<i>collection</i>	collection to be checked for containment in this collection
-------------------	---

#### Returns

true if this collection contains all of the elements in the specified collection.

#### Exceptions

<i>Exception</i>	if an error occurs,
------------------	---------------------

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 165).

```
6.796.3.4 template<typename E > virtual std::size_t
decaf::util::concurrent::SynchronousQueue<
E >::drainTo ( Collection< E > & c ) throw ( de-
caaf::lang::exceptions::UnsupportedOperationException,
decaf::lang::exceptions::IllegalArgumentException ) [inline,
virtual]
```

Removes all available elements from this queue and adds them to the given collection.

This operation may be more efficient than repeatedly polling this queue. A failure encountered while attempting to add elements to collection *c* may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

#### Parameters

<i>c</i>	the collection to transfer elements into
----------	--

#### Returns

the number of elements transferred

#### Exceptions

<i>UnsupportedOperationException</i>	if addition of elements is not supported by the specified collection
<i>IllegalArgumentException</i>	if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 852).

References `decaf::util::AbstractQueue< E >::element()`, and `decaf::util::concurrent::SynchronousQueue< E >::poll()`.

```
6.796.3.5  template<typename E > virtual std::size_t
           decaf::util::concurrent::SynchronousQueue< E
           >::drainTo ( Collection< E > & c, std::size_t maxElements ) throw
           ( decaf::lang::exceptions::UnsupportedOperationException,
           decaf::lang::exceptions::IllegalArgumentException ) [inline,
           virtual]
```

Removes at most the given number of available elements from this queue and adds them to the given collection.

A failure encountered while attempting to add elements to collection `c` may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

#### Parameters

<code>c</code>	the collection to transfer elements into
<code>maxElements</code>	the maximum number of elements to transfer

#### Returns

the number of elements transferred

#### Exceptions

<i>UnsupportedOperationException</i>	if addition of elements is not supported by the specified collection
<i>IllegalArgumentException</i>	if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implements `decaf::util::concurrent::BlockingQueue< E >` (p. 852).

References `decaf::util::AbstractQueue< E >::element()`, and `decaf::util::concurrent::SynchronousQueue< E >::poll()`.

```
6.796.3.6  template<typename E > virtual bool
           decaf::util::concurrent::SynchronousQueue< E
           >::equals ( const Collection< E > & collection ) const [inline,
           virtual]
```

Answers true if this **Collection** (p. 1216) and the one given are the same size and if each element contained in the **Collection** (p. 1216) given is equal to an element contained in this collection.

### Parameters

<i>collection</i>	- The <b>Collection</b> (p. 1216) to be compared to this one.
-------------------	---

### Returns

true if this **Collection** (p. 1216) is equal to the one given.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 166).

```
6.796.3.7  template<typename E > virtual bool  
           decaf::util::concurrent::SynchronousQueue< E  
           >::isEmpty ( ) const [inline, virtual]
```

Returns true if this collection contains no elements.

This implementation returns **size()** (p. 3837) == 0.

### Returns

true if the size method return 0.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 166).

```
6.796.3.8  template<typename E > virtual decaf::util::Iterator<E>*  
           decaf::util::concurrent::SynchronousQueue< E >::iterator ( ) const  
           [inline, virtual]
```

Implements **decaf::lang::Iterable< E >** (p. 2222).

```
6.796.3.9  template<typename E > virtual decaf::util::Iterator<E>*  
           decaf::util::concurrent::SynchronousQueue< E >::iterator ( )  
           [inline, virtual]
```

### Returns

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p. 2221).

```
6.796.3.10 template<typename E > virtual bool  
           decaf::util::concurrent::SynchronousQueue< E  
           >::offer ( const E & e, long timeout, const TimeUnit & unit )  
           throw ( decaf::lang::exceptions::InterruptedException,  
                   decaf::lang::exceptions::NullPointerException,  
                   decaf::lang::exceptions::IllegalArgumentException ) [inline,  
                               virtual]
```

Inserts the specified element into this queue, waiting if necessary up to the specified wait time for another thread to receive it.

**Returns**

`true` if successful, or `false` if the specified waiting time elapses before a consumer appears.

**Exceptions**

<i>InterruptedException</i>	Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.
<i>NullPointerException</i>	Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.
<i>IllegalArgumentException</i>	Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 853).

```
6.796.3.11  template<typename E > virtual bool
             decaf::util::concurrent::SynchronousQueue< E >::offer ( const E
             & value ) throw ( decaf::lang::exceptions::NullPointerException,
             decaf::lang::exceptions::IllegalArgumentException ) [inline,
             virtual]
```

Inserts the specified element into this queue, if another thread is waiting to receive it.

**Parameters**

<i>value</i>	the element to add to the <b>Queue</b> (p. 3239)
--------------	--

**Returns**

`true` if the element was added to this queue, else `false`

**Exceptions**

<i>NullPointerException</i>	if the <b>Queue</b> (p. 3239) implementation does not allow Null values to be inserted into the <b>Queue</b> (p. 3239).
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this queue

Implements **decaf::util::Queue< E >** (p. 3241).

6.796.3.12 `template<typename E > virtual bool  
decaf::util::concurrent::SynchronousQueue< E  
>::peek ( E &result DECAF_UNUSED ) const [inline, virtual]`

6.796.3.13 `template<typename E > virtual bool  
decaf::util::concurrent::SynchronousQueue< E  
>::poll ( E &result, long long timeout, const TimeUnit &unit ) throw (   
decaf::lang::exceptions::InterruptedException ) [inline,  
virtual]`

Retrieves and removes the head of this queue, waiting if necessary up to the specified wait time, for another thread to insert it.

#### Parameters

<i>result</i>	a reference to the value where the head of the <b>Queue</b> (p. 3239) should be copied to.
<i>timeout</i>	the time that the method should block if there is no element available to return.
<i>unit</i>	the Time Units that the timeout value represents.

#### Returns

true if the head of the **Queue** (p. 3239) was copied to the result param or false if no value could be returned.

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 854).

Referenced by **decaf::util::concurrent::SynchronousQueue< E >::drainTo()**.

6.796.3.14 `template<typename E > virtual bool  
decaf::util::concurrent::SynchronousQueue< E  
>::poll ( E &result ) [inline, virtual]`

Retrieves and removes the head of this queue, if another thread is currently making an element available.

#### Parameters

<i>result</i>	a reference to the value where the head of the <b>Queue</b> (p. 3239) should be copied to.
---------------	--

#### Returns

true if the head of the **Queue** (p. 3239) was copied to the result param or false if no value could be returned.

Implements **decaf::util::Queue< E >** (p. 3242).

```

6.796.3.15  template<typename E > virtual void
             decaf::util::concurrent::SynchronousQueue< E >::put
             ( const E & value ) throw ( decaf::lang::exceptions::InterruptedException,
             decaf::lang::exceptions::NullPointerException,
             decaf::lang::exceptions::IllegalArgumentException ) [inline,
             virtual]

```

Adds the specified element to this queue, waiting if necessary for another thread to receive it.

#### Parameters

<i>value</i>	the element to add to the <b>Queue</b> (p. 3239).
--------------	---

#### Exceptions

<i>InterruptedException</i>	Inserts the specified element into this queue, waiting if necessary for space to become available.
<i>NullPointerException</i>	Inserts the specified element into this queue, waiting if necessary for space to become available.
<i>IllegalArgumentException</i>	Inserts the specified element into this queue, waiting if necessary for space to become available.

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 854).

```

6.796.3.16  template<typename E > virtual int
             decaf::util::concurrent::SynchronousQueue< E
             >::remainingCapacity( ) const [inline, virtual]

```

Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or `Integer::MAX_VALUE` if there is no intrinsic limit.

Note that you *cannot* always tell if an attempt to insert an element will succeed by inspecting `remainingCapacity` because it may be the case that another thread is about to insert or remove an element.

#### Returns

the remaining capacity

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 855).



- 6.796.3.17 `template<typename E > virtual bool  
decaf::util::concurrent::SynchronousQueue< E  
>::remove ( const E &value DECAF_UNUSED ) throw (   
lang::exceptions::UnsupportedOperationException,  
lang::exceptions::IllegalArgumentException ) [inline,  
virtual]`
- 6.796.3.18 `template<typename E > virtual bool  
decaf::util::concurrent::SynchronousQueue< E  
>::removeAll ( const Collection< E > &collection DECAF_UNUSED )  
throw ( lang::exceptions::UnsupportedOperationException,  
lang::exceptions::IllegalArgumentException ) [inline,  
virtual]`
- 6.796.3.19 `template<typename E > virtual bool  
decaf::util::concurrent::SynchronousQueue< E  
>::retainAll ( const Collection< E > &collection DECAF_UNUSED )  
throw ( lang::exceptions::UnsupportedOperationException,  
lang::exceptions::IllegalArgumentException ) [inline,  
virtual]`
- 6.796.3.20 `template<typename E > virtual std::size_t  
decaf::util::concurrent::SynchronousQueue< E  
>::size ( ) const [inline, virtual]`

Returns the number of elements in this collection.

If this collection contains more than Integer.MAX\_VALUE elements, returns Integer.MAX\_VALUE.

#### Returns

the number of elements in this collection

Implements **decaf::util::Collection< E >** (p. 1226).

- 6.796.3.21 `template<typename E > virtual E  
decaf::util::concurrent::SynchronousQueue< E  
>::take ( ) throw ( decaf::lang::exceptions::InterruptedException )  
[inline, virtual]`

Retrieves and removes the head of this queue, waiting if necessary for another thread to insert it.

#### Returns

the head of this queue

#### Exceptions

<i>InterruptedException</i>	Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.
-----------------------------	--

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 855).

```
6.796.3.22 template<typename E > virtual std::vector<E>
decaf::util::concurrent::SynchronousQueue< E >::toArray ( ) const
[inline, virtual]
```

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 1216).

All the elements in the array will not be referenced by the collection. The elements in the returned array will be sorted to the same order as those returned by the iterator of this collection itself if the collection guarantees the order.

### Returns

an vector of copies of all the elements from this **Collection** (p. 1216)

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 170).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/SynchronousQueue.h

## 6.797 decaf::lang::System Class Reference

The **System** (p. 3838) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.

```
#include <src/main/decaf/lang/System.h>
```

### Public Member Functions

- virtual **~System** ()

### Static Public Member Functions

- static void **arraycopy** (const unsigned char \*src, std::size\_t srcPos, unsigned char \*dest, std::size\_t destPos, std::size\_t length)

*Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.*

- static void **arraycopy** (const short \*src, std::size\_t srcPos, short \*dest, std::size\_t destPos, std::size\_t length)

*Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.*

- static void **arraycopy** (const int \*src, std::size\_t srcPos, int \*dest, std::size\_t destPos, std::size\_t length)  
*Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.*
- static void **arraycopy** (const long long \*src, std::size\_t srcPos, long long \*dest, std::size\_t destPos, std::size\_t length)  
*Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.*
- static const **util::Map**< std::string, std::string > & **getenv** ()  
*Enumerates the system environment and returns a map of env variable names to the string values they hold.*
- static std::string **getenv** (const std::string &name)  
*Reads an environment value from the system and returns it as a string object.*
- static void **unsetenv** (const std::string &name)  
*Clears a set environment value if one is set.*
- static void **setenv** (const std::string &name, const std::string &value)  
*Sets the specified system property to the value given.*
- static long long **currentTimeMillis** ()  
*Returns the current time in milliseconds.*
- static long long **nanoTime** ()  
*Returns the current value of the most precise available system timer, in nanoseconds.*
- static int **availableProcessors** ()  
*Returns the number of processors available for execution of Decaf Threads.*
- static **decaf::util::Properties** & **getProperties** ()  
*Gets the Properties object that holds the Properties accessed from calls to getProperty and setProperty.*
- static std::string **getProperty** (const std::string &key)  
*Gets the specified **System** (p. 3838) property if set, otherwise returns an empty string.*
- static std::string **getProperty** (const std::string &key, const std::string &default-Value)  
*Gets the specified **System** (p. 3838) property if set, otherwise returns the specified default value.*

- static std::string **setProperty** (const std::string &key, const std::string &value)  
*Sets the **System** (p. 3838) Property to the specified value.*
- static std::string **clearProperty** (const std::string &key)  
*Clear any value associated with the system property specified.*

## Protected Member Functions

- **System** ()

## Friends

- class **decaf::lang::Runtime**

### 6.797.1 Detailed Description

The **System** (p. 3838) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.

#### Since

1.0

### 6.797.2 Constructor & Destructor Documentation

**6.797.2.1** **decaf::lang::System::System** ( ) [protected]

**6.797.2.2** **virtual decaf::lang::System::~~System** ( ) [inline, virtual]

### 6.797.3 Member Function Documentation

**6.797.3.1** **static void decaf::lang::System::arraycopy** ( const unsigned char \* *src*, std::size\_t *srcPos*, unsigned char \* *dest*, std::size\_t *destPos*, std::size\_t *length* ) [static]

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

#### Parameters

<i>src</i>	The source array to copy from.
<i>srcPos</i>	The position in the array to start copying from.
<i>dest</i>	The destination array to copy to.
<i>destPos</i>	The position in the destination array to start writing at.
<i>length</i>	The number of elements to copy from src to dest.

**Exceptions**

<i>NullPointerException</i>	if src or dest are NULL.
-----------------------------	--------------------------

Referenced by decaf::lang::ArrayPointer< unsigned char >::clone().

**6.797.3.2** `static void decaf::lang::System::arraycopy ( const short * src, std::size_t srcPos, short * dest, std::size_t destPos, std::size_t length ) [static]`

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

**Parameters**

<i>src</i>	The source array to copy from.
<i>srcPos</i>	The position in the array to start copying from.
<i>dest</i>	The destination array to copy to.
<i>destPos</i>	The position in the destination array to start writing at.
<i>length</i>	The number of elements to copy from src to dest.

**Exceptions**

<i>NullPointerException</i>	if src or dest are NULL.
-----------------------------	--------------------------

**6.797.3.3** `static void decaf::lang::System::arraycopy ( const long long * src, std::size_t srcPos, long long * dest, std::size_t destPos, std::size_t length ) [static]`

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

**Parameters**

<i>src</i>	The source array to copy from.
<i>srcPos</i>	The position in the array to start copying from.
<i>dest</i>	The destination array to copy to.
<i>destPos</i>	The position in the destination array to start writing at.
<i>length</i>	The number of elements to copy from src to dest.

**Exceptions**

<i>NullPointerException</i>	if src or dest are NULL.
-----------------------------	--------------------------

**6.797.3.4** `static void decaf::lang::System::arraycopy ( const int * src, std::size_t srcPos, int * dest, std::size_t destPos, std::size_t length )` [static]

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

#### Parameters

<i>src</i>	The source array to copy from.
<i>srcPos</i>	The position in the array to start copying from.
<i>dest</i>	The destination array to copy to.
<i>destPos</i>	The position in the destination array to start writing at.
<i>length</i>	The number of elements to copy from src to dest.

#### Exceptions

<i>NullPointerException</i>	if src or dest are NULL.
-----------------------------	--------------------------

**6.797.3.5** `static int decaf::lang::System::availableProcessors ( )` [static]

Returns the number of processors available for execution of Decaf Threads.

This value may change during a particular execution of a Decaf based application. Applications that are sensitive to the number of available processors should therefore occasionally poll this property and adjust their resource usage appropriately.

#### Returns

the number of available processors.

**6.797.3.6** `static std::string decaf::lang::System::clearProperty ( const std::string & key )`  
[static]

Clear any value associated with the system property specified.

#### Parameters

<i>key</i>	The key name of the system property to clear.
------------	---

#### Returns

the previous value of the property named by key if there was one, otherwise returns an empty string.

#### Exceptions

<i>IllegalArgumentException</i>	if key is an empty string.
---------------------------------	----------------------------

**6.797.3.7** `static long long decaf::lang::System::currentTimeMillis ( ) [static]`

Returns the current time in milliseconds.

Note that while the unit of time of the return value is a millisecond, the granularity of the value depends on the underlying operating system and may be larger. For example, many operating systems measure time in units of tens of milliseconds.

See the description of the class Date for a discussion of slight discrepancies that may arise between "computer time" and coordinated universal time (UTC).

#### Returns

the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.

**6.797.3.8** `static const util::Map<std::string, std::string>& decaf::lang::System::getenv ( ) [static]`

Enumerates the system environment and returns a map of env variable names to the string values they hold.

#### Returns

A Map of all environment variables.

#### Exceptions

<i>Exception</i> (p. 1886)	if an error occurs while getting the Environment Map.
----------------------------	---

**6.797.3.9** `static std::string decaf::lang::System::getenv ( const std::string & name ) [static]`

Reads an environment value from the system and returns it as a string object.

#### Parameters

<i>name</i>	The environment variable to read.
-------------	-----------------------------------

#### Returns

a string with the value from the variables or ""

#### Exceptions

<i>an</i>	<b>Exception</b> (p. 1886) if an error occurs while reading the Env.
-----------	--

**6.797.3.10** `static decaf::util::Properties& decaf::lang::System::getProperties ( )`  
`[static]`

Gets the Properties object that holds the Properties accessed from calls to `getProperty` and `setProperty`.

If the Properties has not yet been created or are not yet initialized then they will be on the first call to a Properties accessor.

#### Returns

a reference to the static system Properties object.

**6.797.3.11** `static std::string decaf::lang::System::getProperty ( const std::string & key )`  
`[static]`

Gets the specified **System** (p. 3838) property if set, otherwise returns an empty string.

If the Properties has not yet been created or are not yet initialized then they will be on the first call to a Properties accessor.

#### Parameters

<i>key</i>	The key name of the desired system property to retrieve.
------------	--

#### Returns

an empty string if the named property is not set, otherwise returns the value.

#### Exceptions

<i>IllegalArgumentEx- ception</i>	if key is an empty string.
---------------------------------------	----------------------------

**6.797.3.12** `static std::string decaf::lang::System::getProperty ( const std::string & key, const std::string & defaultValue )` `[static]`

Gets the specified **System** (p. 3838) property if set, otherwise returns the specified default value.

If the Properties has not yet been created or are not yet initialized then they will be on the first call to a Properties accessor.

#### Parameters

<i>key</i>	The key name of the desired system property to retrieve.
<i>defaultValue</i>	The default value to return if the key is not set in the <b>System</b> (p. 3838) properties.



**Returns**

the value of the named system property or the default value if the property isn't set..

**Exceptions**

<i>IllegalArgumentException</i>	if key is an empty string.
---------------------------------	----------------------------

**6.797.3.13 static long long decaf::lang::System::nanoTime ( ) [static]**

Returns the current value of the most precise available system timer, in nanoseconds.

This method can only be used to measure elapsed time and is not related to any other notion of system or wall-clock time. The value returned represents nanoseconds since some fixed but arbitrary time (perhaps in the future, so values may be negative). This method provides nanosecond precision, but not necessarily nanosecond accuracy. No guarantees are made about how frequently values change. Differences in successive calls that span greater than approximately 292 years (263 nanoseconds) will not accurately compute elapsed time due to numerical overflow.

For example, to measure how long some code takes to execute:

```
long long startTime = System::nanoTime() (p. 3845); // ... the code being measured
... long long estimatedTime = System::nanoTime() (p. 3845) - startTime;
```

**Returns**

The current value of the system timer, in nanoseconds.

**6.797.3.14 static void decaf::lang::System::setenv ( const std::string & name, const std::string & value ) [static]**

Sets the specified system property to the value given.

**Parameters**

<i>name</i>	The name of the environment variables to set.
<i>value</i>	The value to assign to name.

**Exceptions**

<i>an</i>	<b>Exception</b> (p. 1886) if an error occurs when setting the environment variable.
-----------	--

**6.797.3.15** `static std::string decaf::lang::System::setProperty ( const std::string & key, const std::string & value )` `[static]`

Sets the **System** (p. 3838) Property to the specified value.

#### Parameters

<i>key</i>	The key name of the system property to set to the given value.
<i>value</i>	The value to assign to the key.

#### Returns

the previous value of the property named by key if there was one, otherwise returns an empty string.

#### Exceptions

<i>IllegalArgumentException</i>	if key is an empty string.
---------------------------------	----------------------------

**6.797.3.16** `static void decaf::lang::System::unsetenv ( const std::string & name )` `[static]`

Clears a set environment value if one is set.

#### Parameters

<i>name</i>	The environment variables to clear.
-------------	-------------------------------------

#### Exceptions

<i>an</i>	<b>Exception</b> (p. 1886) if an error occurs while reading the environment.
-----------	--

## 6.797.4 Friends And Related Function Documentation

**6.797.4.1** `friend class decaf::lang::Runtime` `[friend]`

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/System.h`

## 6.798 activemq::threads::Task Class Reference

Represents a unit of work that requires one or more iterations to complete.

```
#include <src/main/activemq/threads/Task.h>
```

Inheritance diagram for `activemq::threads::Task`:

## Public Member Functions

- virtual `~Task()`
- virtual `bool iterate()=0`

*Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.*

### 6.798.1 Detailed Description

Represents a unit of work that requires one or more iterations to complete.

Since

3.0

### 6.798.2 Constructor & Destructor Documentation

6.798.2.1 virtual `activemq::threads::Task::~~Task()` [`inline`, `virtual`]

### 6.798.3 Member Function Documentation

6.798.3.1 virtual `bool activemq::threads::Task::iterate()` [`pure virtual`]

Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

**Returns**

true if the task should be run again or false if the task has completed and the runner should wait for a wakeup call.

Implemented in `activemq::core::ActiveMQSessionExecutor` (p. 535), `activemq::threads::CompositeTaskRunner` (p. 1257), `activemq::transport::failover::BackupTransportPool` (p. 764), `activemq::transport::failover::CloseTransportPool` (p. 1183), and `activemq::transport::failover::FailoverTransport` (p. 1937).

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/Task.h`

## 6.799 decaf::util::concurrent::TaskListener Class Reference

```
#include <src/main/decaf/util/concurrent/TaskListener.h>
```

## Public Member Functions

- virtual `~TaskListener ()`
- virtual void `onTaskComplete (lang::Runnable *task)=0`  
*Called when a queued task has completed, the task that finished is passed along for user consumption.*
- virtual void `onTaskException (lang::Runnable *task, lang::Exception &ex)=0`  
*Called when a queued task has thrown an exception while being run.*

## 6.799.1 Constructor & Destructor Documentation

- 6.799.1.1 `virtual decaf::util::concurrent::TaskListener::~~TaskListener ( ) [inline, virtual]`

## 6.799.2 Member Function Documentation

- 6.799.2.1 `virtual void decaf::util::concurrent::TaskListener::onTaskComplete ( lang::Runnable * task ) [pure virtual]`

Called when a queued task has completed, the task that finished is passed along for user consumption.

### Parameters

<i>task</i>	Runnable Pointer to the task that finished
-------------	--

- 6.799.2.2 `virtual void decaf::util::concurrent::TaskListener::onTaskException ( lang::Runnable * task, lang::Exception & ex ) [pure virtual]`

Called when a queued task has thrown an exception while being run.

The Callee should assume that this was an unrecoverable exception and that this task is now defunct.

### Parameters

<i>task</i>	Runnable Pointer to the task
<i>ex</i>	The ActiveMQException that was thrown.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/TaskListener.h`

## 6.800 activemq::threads::TaskRunner Class Reference

```
#include <src/main/activemq/threads/TaskRunner.h>
```

Inheritance diagram for activemq::threads::TaskRunner:

### Public Member Functions

- virtual **~TaskRunner** ()
- virtual void **shutdown** (unsigned int timeout)=0  
*Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.*
- virtual void **shutdown** ()=0  
*Shutdown once the task has finished and the TaskRunner's thread has exited.*
- virtual void **wakeup** ()=0  
*Signal the **TaskRunner** (p. 3849) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3846) instance will be run until its iterate method has returned false indicating it is done.*

### 6.800.1 Constructor & Destructor Documentation

**6.800.1.1** virtual activemq::threads::TaskRunner::~TaskRunner ( ) [inline, virtual]

### 6.800.2 Member Function Documentation

**6.800.2.1** virtual void activemq::threads::TaskRunner::shutdown ( unsigned int *timeout* ) [pure virtual]

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

#### Parameters

<i>timeout</i>	- Time in Milliseconds to wait for the task to stop.
----------------	--

Implemented in **activemq::threads::CompositeTaskRunner** (p. 1258), and **activemq::threads::DedicatedTaskRunner** (p. 1726).

**6.800.2.2** virtual void activemq::threads::TaskRunner::shutdown ( ) [pure virtual]

Shutdown once the task has finished and the TaskRunner's thread has exited.

Implemented in `activemq::threads::CompositeTaskRunner` (p. 1258), and `activemq::threads::DedicatedTaskRunner` (p. 1725).

**6.800.2.3** `virtual void activemq::threads::TaskRunner::wakeup ( )` [pure virtual]

Signal the **TaskRunner** (p. 3849) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3846) instance will be run until its `iterate` method has returned false indicating it is done.

Implemented in `activemq::threads::CompositeTaskRunner` (p. 1258), and `activemq::threads::DedicatedTaskRunner` (p. 1726).

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/TaskRunner.h`

## 6.801 decaf::internal::net::tcp::TcpSocket Class Reference

Platform-independent implementation of the socket interface.

```
#include <src/main/decaf/internal/net/tcp/TcpSocket.h>
```

Inheritance diagram for `decaf::internal::net::tcp::TcpSocket`:

### Public Member Functions

- **TcpSocket** () throw ( `decaf::net::SocketException` )  
*Construct a non-connected socket.*
- virtual **~TcpSocket** ()  
*Releases the socket handle but not gracefully shut down the connection.*
- SocketHandle **getSocketHandle** ()  
*Gets the handle for the socket.*
- bool **isConnected** () const
- bool **isClosed** () const
- virtual std::string **getLocalAddress** () const  
*Gets the value of the local Inet address the **Socket** (p. 3607) is bound to if bound, otherwise return the **InetAddress** (p. 2077) ANY value "0.0.0.0".*  
**Returns**  
*the local address bound to, or ANY.*
- virtual void **create** () throw ( `decaf::io::IOException` )

Creates the underlying platform **Socket** (p. 3607) data structures which allows for **Socket** (p. 3607) options to be applied.

The created socket is in an unconnected state.

#### Exceptions

IOException	if an I/O error occurs while attempting this operation.
-------------	---

- virtual void **accept** (SocketImpl \*socket) throw ( decaf::io::IOException )
- virtual void **bind** (const std::string &ipaddress, int **port**) throw ( decaf::io::IOException )

Binds this **Socket** (p. 3607) instance to the local ip address and port number given.

#### Parameters

ipaddress	The address of local ip to bind to.
port	The port number on the host to bind to.

#### Exceptions

IOException	if an I/O error occurs while attempting this operation.
-------------	---

- virtual void **connect** (const std::string &hostname, int **port**, int timeout) throw ( decaf::io::IOException, decaf::net::SocketException, decaf::lang::exceptions::IllegalArgumentException )

Connects this socket to the given host and port.

#### Parameters

hostname	The name of the host to connect to, or IP address.
port	The port number on the host to connect to.
timeout	Time in milliseconds to wait for a connection, 0 indicates forever.

#### Exceptions

IOException	if an I/O error occurs while attempting this operation.
<b>SocketTimeoutException</b> (p. 3650)	if the connect call times out due to timeout being set.
IllegalArgumentEx-ception	if a parameter has an illegal value.

- virtual void **listen** (int backlog) throw ( decaf::io::IOException )

Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.

If a connection indication arrives when the queue is full, the connection is refused.

#### Parameters

backlog	The maximum length of the connection queue.
---------	---

#### Exceptions

IOException	if an I/O error occurs while attempting this operation.
-------------	---

- virtual **decaf::io::InputStream \* getInputStream ()** throw ( decaf::io::IOException )

*Gets the InputStream linked to this **Socket** (p. 3607).*

**Returns**

*an InputStream pointer owned by the **Socket** (p. 3607) object.*

**Exceptions**

IOException	<i>if an I/O error occurs while attempting this operation.</i>
-------------	--

- virtual **decaf::io::OutputStream \* getOutputStream ()** throw ( decaf::io::IOException )

*Gets the OutputStream linked to this **Socket** (p. 3607).*

**Returns**

*an OutputStream pointer owned by the **Socket** (p. 3607) object.*

**Exceptions**

IOException	<i>if an I/O error occurs while attempting this operation.</i>
-------------	--

- virtual int **available ()** throw ( decaf::io::IOException )

*Gets the number of bytes that can be read from the **Socket** (p. 3607) without blocking.*

**Returns**

*the number of bytes that can be read from the **Socket** (p. 3607) without blocking.*

**Exceptions**

IOException	<i>if an I/O error occurs while attempting this operation.</i>
-------------	--

- virtual void **close ()** throw ( decaf::io::IOException )

*Closes the socket, terminating any blocked reads or writes.*

**Exceptions**

IOException	<i>if an I/O error occurs while attempting this operation.</i>
-------------	--

- virtual void **shutdownInput ()** throw ( decaf::io::IOException )

*Places the input stream for this socket at "end of stream".*

*Any data sent to this socket is acknowledged and then silently discarded. If you read from a socket input stream after invoking **shutdownInput()** (p. 3643) on the socket, the stream will return EOF.*

**Exceptions**

IOException	<i>if an I/O error occurs while attempting this operation.</i>
-------------	--

- virtual void **shutdownOutput ()** throw ( decaf::io::IOException )

*Disables the output stream for this socket.*

*For a TCP socket, any previously written data will be sent followed by TCP's normal connection termination sequence. If you write to a socket output stream after invoking*



*shutdownOutput()* (p. 3643) on the socket, the stream will throw an *IOException*.

#### Exceptions

IOException	if an I/O error occurs while attempting this operation.
-------------	---

- virtual int **getOption** (int option) const throw ( decaf::io::IOException )

*Gets the specified **Socket** (p. 3607) option.*

#### Parameters

option	The <b>Socket</b> (p. 3607) options whose value is to be retrieved.
--------	---

#### Returns

*the value of the given socket option.*

#### Exceptions

IOException	if an I/O error occurs while performing this operation.
-------------	---

- virtual void **setOption** (int option, int value) throw ( decaf::io::IOException )

*Sets the specified option on the **Socket** (p. 3607) if supported.*

#### Parameters

option	The <b>Socket</b> (p. 3607) option to set.
value	The value of the socket option to apply to the socket.

#### Exceptions

IOException	if an I/O error occurs while performing this operation.
-------------	---

- int **read** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*Reads the requested data from the Socket and write it into the passed in buffer.*

- void **write** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*Writes the specified data in the passed in buffer to the Socket.*

## Protected Member Functions

- void **checkResult** (apr\_status\_t value) const throw ( decaf::net::SocketException )

### 6.801.1 Detailed Description

Platform-independent implementation of the socket interface.

## 6.801.2 Constructor & Destructor Documentation

6.801.2.1 `decaf::internal::net::tcp::TcpSocket::TcpSocket ( ) throw ( decaf::net::SocketException )`

Construct a non-connected socket.

### Exceptions

<i>SocketException</i>	thrown if an error occurs while creating the Socket.
------------------------	--

6.801.2.2 `virtual decaf::internal::net::tcp::TcpSocket::~~TcpSocket ( ) [virtual]`

Releases the socket handle but not gracefully shut down the connection.

## 6.801.3 Member Function Documentation

6.801.3.1 `virtual void decaf::internal::net::tcp::TcpSocket::accept ( SocketImpl * socket ) throw ( decaf::io::IOException ) [virtual]`

6.801.3.2 `virtual int decaf::internal::net::tcp::TcpSocket::available ( ) throw ( decaf::io::IOException ) [virtual]`

Gets the number of bytes that can be read from the **Socket** (p. 3607) without blocking.

### Returns

the number of bytes that can be read from the **Socket** (p. 3607) without blocking.

### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3638).

6.801.3.3 `virtual void decaf::internal::net::tcp::TcpSocket::bind ( const std::string & ipaddress, int port ) throw ( decaf::io::IOException ) [virtual]`

Binds this **Socket** (p. 3607) instance to the local ip address and port number given.

### Parameters

<i>ipaddress</i>	The address of local ip to bind to.
<i>port</i>	The port number on the host to bind to.

### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3638).

**6.801.3.4** `void decaf::internal::net::tcp::TcpSocket::checkResult ( apr_status_t value ) const  
throw ( decaf::net::SocketException )` [protected]

**6.801.3.5** `virtual void decaf::internal::net::tcp::TcpSocket::close ( ) throw ( decaf::io::IOException )` [virtual]

Closes the socket, terminating any blocked reads or writes.

### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3639).

**6.801.3.6** `virtual void decaf::internal::net::tcp::TcpSocket::connect ( const std::string & hostname, int port, int timeout ) throw ( decaf::io::IOException, decaf::net::SocketException, decaf::lang::exceptions::IllegalArgumentException )` [virtual]

Connects this socket to the given host and port.

### Parameters

<i>hostname</i>	The name of the host to connect to, or IP address.
<i>port</i>	The port number on the host to connect to.
<i>timeout</i>	Time in milliseconds to wait for a connection, 0 indicates forever.

### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
<i>SocketTimeoutException</i> (p. 3650)	if the connect call times out due to timeout being set.
<i>IllegalArgumentEx-ception</i>	if a parameter has an illegal value.

Implements **decaf::net::SocketImpl** (p. 3639).

**6.801.3.7** `virtual void decaf::internal::net::tcp::TcpSocket::create ( ) throw ( decaf::io::IOException )` [virtual]

Creates the underlying platform **Socket** (p. 3607) data structures which allows for **Socket** (p. 3607) options to be applied.

The created socket is in an unconnected state.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3640).

**6.801.3.8** `virtual decaf::io::InputStream* decaf::internal::net::tcp::TcpSocket::getInputStream ( ) throw ( decaf::io::IOException )` [virtual]

Gets the InputStream linked to this **Socket** (p. 3607).

**Returns**

an InputStream pointer owned by the **Socket** (p. 3607) object.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3640).

**6.801.3.9** `virtual std::string decaf::internal::net::tcp::TcpSocket::getLocalAddress ( ) const` [virtual]

Gets the value of the local Inet address the **Socket** (p. 3607) is bound to if bound, otherwise return the **InetAddress** (p. 2077) ANY value "0.0.0.0".

**Returns**

the local address bound to, or ANY.

Implements **decaf::net::SocketImpl** (p. 3641).

**6.801.3.10** `virtual int decaf::internal::net::tcp::TcpSocket::getOption ( int option ) const throw ( decaf::io::IOException )` [virtual]

Gets the specified **Socket** (p. 3607) option.

**Parameters**

<i>option</i>	The <b>Socket</b> (p. 3607) options whose value is to be retrieved.
---------------	---

**Returns**

the value of the given socket option.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3641).

**6.801.3.11** virtual **decaf::io::OutputStream\*** **decaf::internal::net::tcp::TcpSocket::getOutputStream** ( ) throw ( **decaf::io::IOException** ) [virtual]

Gets the OutputStream linked to this **Socket** (p. 3607).

#### Returns

an OutputStream pointer owned by the **Socket** (p. 3607) object.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3641).

**6.801.3.12** **SocketHandle** **decaf::internal::net::tcp::TcpSocket::getSocketHandle** ( ) [inline]

Gets the handle for the socket.

#### Returns

SocketHabler for this Socket, can be NULL

**6.801.3.13** **bool** **decaf::internal::net::tcp::TcpSocket::isClosed** ( ) const [inline]

#### Returns

true if the close method has been called on this Socket.

**6.801.3.14** **bool** **decaf::internal::net::tcp::TcpSocket::isConnected** ( ) const [inline]

#### Returns

true if the socketHandle is not in a disconnected state.

**6.801.3.15** virtual void **decaf::internal::net::tcp::TcpSocket::listen** ( int *backlog* ) throw ( **decaf::io::IOException** ) [virtual]

Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.

If a connection indication arrives when the queue is full, the connection is refused.

**Parameters**

<i>backlog</i>	The maximum length of the connection queue.
----------------	---

**Exceptions**

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3642).

**6.801.3.16** `int decaf::internal::net::tcp::TcpSocket::read ( unsigned char * buffer,  
int size, int offset, int length ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::NullPointerException )`

Reads the requested data from the Socket and write it into the passed in buffer.

**Parameters**

<i>buffer</i>	The buffer to read into
<i>size</i>	The size of the specified buffer
<i>offset</i>	The offset into the buffer where reading should start filling.
<i>length</i>	The number of bytes past offset to fill with data.

**Returns**

the actual number of bytes read or -1 if at EOF.

**Exceptions**

<i>IOException</i>	if an I/O error occurs during the read.
<i>NullPointerException</i>	if buffer is Null.
<i>IndexOutOfBoundsException</i>	if offset + length is greater than buffer size.

**6.801.3.17** `virtual void decaf::internal::net::tcp::TcpSocket::setOption ( int option, int value )  
throw ( decaf::io::IOException )` [virtual]

Sets the specified option on the **Socket** (p. 3607) if supported.

**Parameters**

<i>option</i>	The <b>Socket</b> (p. 3607) option to set.
<i>value</i>	The value of the socket option to apply to the socket.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3643).

**6.801.3.18** virtual void decaf::internal::net::tcp::TcpSocket::shutdownInput ( ) throw ( decaf::io::IOException ) [virtual]

Places the input stream for this socket at "end of stream".

Any data sent to this socket is acknowledged and then silently discarded. If you read from a socket input stream after invoking **shutdownInput()** (p. 3643) on the socket, the stream will return EOF.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3643).

**6.801.3.19** virtual void decaf::internal::net::tcp::TcpSocket::shutdownOutput ( ) throw ( decaf::io::IOException ) [virtual]

Disables the output stream for this socket.

For a TCP socket, any previously written data will be sent followed by TCP's normal connection termination sequence. If you write to a socket output stream after invoking **shutdownOutput()** (p. 3643) on the socket, the stream will throw an IOException.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3643).

**6.801.3.20** void decaf::internal::net::tcp::TcpSocket::write ( const unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

Writes the specified data in the passed in buffer to the Socket.

#### Parameters

<i>buffer</i>	The buffer to write to the socket.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset into the buffer where the data to write starts at.
<i>length</i>	The number of bytes past offset to write.

#### Exceptions

<i>IOException</i>	if an I/O error occurs during the write.
--------------------	--

<i>NullPointerException</i>	if buffer is Null.
<i>IndexOutOfBoundsException</i>	if offset + length is greater than buffer size.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/tcp/TcpSocket.h

## 6.802 decaf::internal::net::tcp::TcpSocketInputStream Class Reference

Input stream for performing reads on a socket.

```
#include <src/main/decaf/internal/net/tcp/TcpSocketInputStream.h>
```

Inheritance diagram for decaf::internal::net::tcp::TcpSocketInputStream:

### Public Member Functions

- **TcpSocketInputStream** (**TcpSocket** \*socket)

*Create a new InputStream to use for reading from the TCP/IP socket.*

- virtual ~**TcpSocketInputStream** ()

- virtual int **available** () const throw ( decaf::io::IOException )

*Indicates the number of bytes available.*

*The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.*

*The default implementation of this method returns zero.*

#### Returns

*the number of bytes available on this input stream.*

#### Exceptions

<b>IOException</b> (p. 2210)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

- virtual void **close** () throw ( decaf::io::IOException )

*Close - does nothing.*

- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::Unsupported )

*Not supported.*



## Protected Member Functions

- virtual int **doReadByte** () throw ( io::IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

### 6.802.1 Detailed Description

Input stream for performing reads on a socket. This class will only work properly for blocking sockets.

Since

1.0

### 6.802.2 Constructor & Destructor Documentation

#### 6.802.2.1 decaf::internal::net::tcp::TcpSocketInputStream::TcpSocketInputStream ( TcpSocket \* *socket* )

Create a new InputStream to use for reading from the TCP/IP socket.

#### Parameters

<i>socket</i>	The parent SocketImpl for this stream.
---------------	--

#### 6.802.2.2 virtual decaf::internal::net::tcp::TcpSocketInputStream::~~TcpSocketInputStream ( ) [virtual]

### 6.802.3 Member Function Documentation

#### 6.802.3.1 virtual int decaf::internal::net::tcp::TcpSocketInputStream::available ( ) const throw ( decaf::io::IOException ) [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

**Returns**

the number of bytes available on this input stream.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
--	-------------------------

Reimplemented from **decaf::io::InputStream** (p. 2107).

**6.802.3.2** `virtual void decaf::internal::net::tcp::TcpSocketInputStream::close ( ) throw ( decaf::io::IOException ) [virtual]`

Close - does nothing.

It is the responsibility of the owner of the socket object to close it.

Closes the **InputStream** (p. 2105) freeing any resources that might have been aquired during the lifetime of this stream.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::InputStream** (p. 2108).

**6.802.3.3** `virtual int decaf::internal::net::tcp::TcpSocketInputStream::doReadArrayBounded ( unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException ) [protected, virtual]`

Reimplemented from **decaf::io::InputStream** (p. 2108).

**6.802.3.4** `virtual int decaf::internal::net::tcp::TcpSocketInputStream::doReadByte ( ) throw ( io::IOException ) [protected, virtual]`

Implements **decaf::io::InputStream** (p. 2109).

**6.802.3.5** `virtual long long decaf::internal::net::tcp::TcpSocketInputStream::skip ( long long num ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Not supported.

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions;

## 6.803 decaf::internal::net::tcp::TcpSocketOutputStream Class Reference 3867

reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The `skip` method of **InputStream** (p. 2105) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

### Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

### Returns

total bytes skipped

### Exceptions

<b>IOException</b> (p. 2210)	if an I/O error occurs.
<b>UnsupportedOperationException</b>	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 2113).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/tcp/TcpSocketInputStream.h`

## 6.803 decaf::internal::net::tcp::TcpSocketOutputStream Class Reference

Output stream for performing write operations on a socket.

```
#include <src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h>
```

Inheritance diagram for `decaf::internal::net::tcp::TcpSocketOutputStream`:

### Public Member Functions

- **TcpSocketOutputStream** (**TcpSocket** \*socket)  
*Create a new instance of a Socket OutputStream class.*
- virtual **~TcpSocketOutputStream** ()
- virtual void **close** () throw ( `decaf::io::IOException` )  
*Closes this object and deallocates the appropriate resources.  
The object is generally no longer usable after calling close.*  
**Exceptions**

<b>IOException</b> (p. 2210)	<i>if an error occurs while closing.</i>
------------------------------	--

*The default implementation of this method does nothing.*

## Protected Member Functions

- virtual void **doWriteByte** (unsigned char c) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### 6.803.1 Detailed Description

Output stream for performing write operations on a socket.

Since

1.0

### 6.803.2 Constructor & Destructor Documentation

6.803.2.1 **decaf::internal::net::tcp::TcpSocketOutputStream::TcpSocketOutputStream ( TcpSocket \* socket )**

Create a new instance of a Socket OutputStream class.

Parameters

<i>socket</i>	The socket to use to write out the data.
---------------	--

6.803.2.2 **virtual decaf::internal::net::tcp::TcpSocketOutputStream::~~TcpSocketOutputStream ( )** [virtual]

### 6.803.3 Member Function Documentation

6.803.3.1 **virtual void decaf::internal::net::tcp::TcpSocketOutputStream::close ( )** throw ( decaf::io::IOException ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

<b>IOException</b> (p. 2210)	if an error occurs while closing.
---------------------------------	-----------------------------------

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2993).

```
6.803.3.2 virtual void decaf::internal::net::tcp::TcpSocketOutputStream::doWriteArrayBounded
( const unsigned char * buffer, int size, int offset, int length ) throw (
  decaf::io::IOException, decaf::lang::exceptions::NullPointerException,
  decaf::lang::exceptions::IndexOutOfBoundsException )
[protected, virtual]
```

Reimplemented from **decaf::io::OutputStream** (p. 2994).

```
6.803.3.3 virtual void decaf::internal::net::tcp::TcpSocketOutputStream::doWriteByte ( unsigned
char c ) throw ( decaf::io::IOException ) [protected, virtual]
```

Implements **decaf::io::OutputStream** (p. 2994).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h

## 6.804 activemq::transport::tcp::TcpTransport Class Reference

Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 2213).

```
#include <src/main/activemq/transport/tcp/TcpTransport.h>
```

Inheritance diagram for activemq::transport::tcp::TcpTransport:

### Public Member Functions

- **TcpTransport** (const **Pointer**< **Transport** > &next)  
*Creates a new instance of a **TcpTransport** (p. 3865), the transport is left unconnected and is in a unusable state until the connect method is called.*
- virtual ~**TcpTransport** ()
- void **connect** (const **decaf::net::URI** &uri, const **decaf::util::Properties** &properties)  
*Creates a Socket and configures it before attempting to connect to the location specified by the URI passed in.*
- virtual void **close** () throw ( decaf::io::IOException )  
*Delegates to the superclass and then closes the socket.*

- virtual bool **isFaultTolerant** () const

*Is this **Transport** (p. 3996) fault tolerant, meaning that it will reconnect to a broker on disconnect.*

- virtual bool **isConnected** () const

*Is the **Transport** (p. 3996) Connected to its Broker.*

- virtual bool **isClosed** () const

*Has the **Transport** (p. 3996) been shutdown and no longer usable.*

## Protected Member Functions

- virtual **decaf::net::Socket \* createSocket** ()

*Create an unconnected Socket instance to be used by the transport to communicate with the broker.*

- virtual void **configureSocket** (**decaf::net::Socket \*socket**, const **decaf::util::Properties &properties**)

*Using options from configuration URI, configure the socket options before the Socket instance is connected to the Server.*

## 6.804.1 Detailed Description

Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 2213). The lower level transport should take care of managing stream reads and writes.

## 6.804.2 Constructor & Destructor Documentation

### 6.804.2.1 **activemq::transport::tcp::TcpTransport::TcpTransport ( const Pointer< Transport > & next )**

Creates a new instance of a **TcpTransport** (p. 3865), the transport is left unconnected and is in a unusable state until the connect method is called.

#### Parameters

<i>next</i>	The next transport in the chain
-------------	---------------------------------

6.804.2.2 virtual `activemq::transport::tcp::TcpTransport::~~TcpTransport ( )` [virtual]

### 6.804.3 Member Function Documentation

6.804.3.1 virtual void `activemq::transport::tcp::TcpTransport::close ( )` throw (`decaf::io::IOException`) [virtual]

Delegates to the superclass and then closes the socket.

#### Exceptions

<i>IOException</i>	if errors occur.
--------------------	------------------

Reimplemented from `activemq::transport::TransportFilter` (p. 4007).

6.804.3.2 virtual void `activemq::transport::tcp::TcpTransport::configureSocket ( decaf::net::Socket * socket, const decaf::util::Properties & properties )` [protected, virtual]

Using options from configuration URI, configure the socket options before the Socket instance is connected to the Server.

Subclasses can override this option to set more configuration options, they should called the base class version to allow the default set of Socket options to also be configured.

#### Parameters

<i>socket</i>	The Socket instance to configure using options from the given Properties.
---------------	---

#### Exceptions

<i>NullPointerException</i>	if the Socket instance is null.
<i>IllegalArgumentException</i>	if the socket instance is not handled by the class.
<i>SocketException</i>	if there is an error while setting one of the Socket options.

6.804.3.3 void `activemq::transport::tcp::TcpTransport::connect ( const decaf::net::URI & uri, const decaf::util::Properties & properties )`

Creates a Socket and configures it before attempting to connect to the location specified by the URI passed in.

The Socket is configured using parameters in the properties that are passed to this method.

#### Parameters

<i>uri</i>	The URI that the <b>Transport</b> (p. 3996) is to connect to once initialized.
------------	--

<i>properties</i>	The Properties that have been parsed from the URI or from configuration files.
-------------------	--

**6.804.3.4** `virtual decaf::net::Socket* activemq::transport::tcp::TcpTransport::createSocket ( )` [protected, virtual]

Create an unconnected Socket instance to be used by the transport to communicate with the broker.

#### Returns

a newly created unconnected Socket instance.

#### Exceptions

<i>IOException</i>	if there is an error while creating the unconnected Socket.
--------------------	---

Reimplemented in `activemq::transport::tcp::SslTransport` (p. 3683).

**6.804.3.5** `virtual bool activemq::transport::tcp::TcpTransport::isClosed ( )` const [inline, virtual]

Has the **Transport** (p. 3996) been shutdown and no longer usable.

#### Returns

true if the **Transport** (p. 3996)

Reimplemented from `activemq::transport::TransportFilter` (p. 4008).

**6.804.3.6** `virtual bool activemq::transport::tcp::TcpTransport::isConnected ( )` const [inline, virtual]

Is the **Transport** (p. 3996) Connected to its Broker.

#### Returns

true if a connection has been made.

Reimplemented from `activemq::transport::TransportFilter` (p. 4008).

**6.804.3.7** `virtual bool activemq::transport::tcp::TcpTransport::isFaultTolerant ( )` const [inline, virtual]

Is this **Transport** (p. 3996) fault tolerant, meaning that it will reconnect to a broker on disconnect.



**Returns**

true if the **Transport** (p. 3996) is fault tolerant.

Reimplemented from `activemq::transport::TransportFilter` (p. 4009).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/tcp/TcpTransport.h`

**6.805 `activemq::transport::tcp::TcpTransportFactory` Class Reference**

Factory Responsible for creating the **TcpTransport** (p. 3865).

```
#include <src/main/activemq/transport/tcp/TcpTransportFactory.h>
```

Inheritance diagram for `activemq::transport::tcp::TcpTransportFactory`:

**Public Member Functions**

- virtual `~TcpTransportFactory()`
- virtual `Pointer< Transport > create` (const `decaf::net::URI` &location) throw ( exceptions::ActiveMQException )  
*Creates a fully configured **Transport** (p. 3996) instance which could be a chain of filters and transports.*
- virtual `Pointer< Transport > createComposite` (const `decaf::net::URI` &location) throw ( exceptions::ActiveMQException )  
*Creates a slimed down **Transport** (p. 3996) instance which can be used in composite transport instances.*

**Protected Member Functions**

- virtual `Pointer< Transport > doCreateComposite` (const `decaf::net::URI` &location, const `Pointer< wireformat::WireFormat >` &wireFormat, const `decaf::util::Properties` &properties) throw ( exceptions::ActiveMQException )  
*Creates a slimed down **Transport** (p. 3996) instance which can be used in composite transport instances.*

**6.805.1 Detailed Description**

Factory Responsible for creating the **TcpTransport** (p. 3865).

## 6.805.2 Constructor & Destructor Documentation

6.805.2.1 `virtual activemq::transport::tcp::TcpTransportFactory::~~TcpTransportFactory ( )`  
`[inline, virtual]`

## 6.805.3 Member Function Documentation

6.805.3.1 `virtual Pointer<Transport> activemq::transport::tcp::TcpTransportFactory::create ( const decaf::net::URI & location ) throw ( exceptions::ActiveMQException )`  
`[virtual]`

Creates a fully configured **Transport** (p.3996) instance which could be a chain of filters and transports.

### Parameters

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

### Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implements `activemq::transport::TransportFactory` (p.4003).

6.805.3.2 `virtual Pointer<Transport> activemq::transport::tcp::TcpTransportFactory::createComposite ( const decaf::net::URI & location ) throw ( exceptions::ActiveMQException )`  
`[virtual]`

Creates a slimed down **Transport** (p.3996) instance which can be used in composite transport instances.

### Parameters

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

### Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implements `activemq::transport::TransportFactory` (p.4004).

```
6.805.3.3 virtual Pointer<Transport> activemq::transport::tcp::TcpTransportFactory::doCreateComposite
( const decaf::net::URI & location, const Pointer<
wireformat::WireFormat > & wireFormat, const decaf::util::Properties &
properties ) throw ( exceptions::ActiveMQException ) [protected,
virtual]
```

Creates a slimmed down **Transport** (p. 3996) instance which can be used in composite transport instances.

#### Parameters

<i>location</i>	- URI location to connect to.
<i>wireFormat</i>	- the assigned WireFormat for the new <b>Transport</b> (p. 3996).
<i>properties</i>	- Properties to apply to the transport.

#### Returns

new Pointer to a **TcpTransport** (p. 3865).

#### Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Reimplemented in **activemq::transport::tcp::SslTransportFactory** (p. 3685).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/tcp/**TcpTransportFactory.h**

## 6.806 cms::TemporaryQueue Class Reference

Defines a Temporary **Queue** (p. 3238) based **Destination** (p. 1776).

```
#include <src/main/cms/TemporaryQueue.h>
```

Inheritance diagram for cms::TemporaryQueue:

#### Public Member Functions

- virtual **~TemporaryQueue** ()
- virtual std::string **getQueueName** () const =0 throw ( CMSEException )  
*Gets the name of this queue.*
- virtual void **destroy** ()=0 throw ( CMSEException )  
*Destroy's the Temporary **Destination** (p. 1776) at the Provider.*

### 6.806.1 Detailed Description

Defines a Temporary **Queue** (p. 3238) based **Destination** (p. 1776). A **TemporaryQueue** (p. 3871) is a special type of **Queue** (p. 3238) **Destination** (p. 1776) that can only be consumed from the **Connection** (p. 1296) which created it. TemporaryQueues are most commonly used as the reply to address for Message's that implement the request response pattern.

A **TemporaryQueue** (p. 3871) is guaranteed to exist at the Provider only for the life-time of the **Connection** (p. 1296) that created it.

#### Since

1.0

### 6.806.2 Constructor & Destructor Documentation

6.806.2.1 `virtual cms::TemporaryQueue::~TemporaryQueue ( ) [inline, virtual]`

### 6.806.3 Member Function Documentation

6.806.3.1 `virtual void cms::TemporaryQueue::destroy ( ) throw ( CMSException ) [pure virtual]`

Destroy's the Temporary **Destination** (p. 1776) at the Provider.

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if an internal error occurs.
---	--------------------------------

Implemented in `activemq::commands::ActiveMQTempQueue` (p. 609).

6.806.3.2 `virtual std::string cms::TemporaryQueue::getQueueName ( ) const throw ( CMSException ) [pure virtual]`

Gets the name of this queue.

#### Returns

The queue name.

#### Exceptions

<b><i>CMSException</i></b> (p. 1190)	- if an internal error occurs.
---	--------------------------------

Implemented in `activemq::commands::ActiveMQTempQueue` (p. 610).

The documentation for this class was generated from the following file:

- src/main/cms/TemporaryQueue.h

## 6.807 cms::TemporaryTopic Class Reference

Defines a Temporary **Topic** (p. 3930) based **Destination** (p. 1776).

```
#include <src/main/cms/TemporaryTopic.h>
```

Inheritance diagram for cms::TemporaryTopic:

### Public Member Functions

- virtual **~TemporaryTopic** ()
- virtual std::string **getTopicName** () const =0 throw ( CMSEException )  
*Gets the name of this topic.*
- virtual void **destroy** ()=0 throw ( CMSEException )  
*Destroy's the Temporary **Destination** (p. 1776) at the Provider.*

### 6.807.1 Detailed Description

Defines a Temporary **Topic** (p. 3930) based **Destination** (p. 1776). A **Temporary-Topic** (p. 3873) is a special type of **Topic** (p. 3930) **Destination** (p. 1776) that can only be consumed from the **Connection** (p. 1296) which created it. TemporaryTopics are most commonly used as the reply to address for Message's that implement the request response pattern.

A **TemporaryTopic** (p. 3873) is guaranteed to exist at the Provider only for the lifetime of the **Connection** (p. 1296) that created it.

Since

1.0

### 6.807.2 Constructor & Destructor Documentation

6.807.2.1 virtual cms::TemporaryTopic::~TemporaryTopic ( ) [inline, virtual]

### 6.807.3 Member Function Documentation

6.807.3.1 virtual void cms::TemporaryTopic::destroy ( ) throw ( CMSEException ) [pure virtual]

Destroy's the Temporary **Destination** (p. 1776) at the Provider.

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)
--

Implemented in **activemq::commands::ActiveMQTempTopic** (p. 639).

**6.807.3.2** `virtual std::string cms::TemporaryTopic::getTopicName ( ) const throw ( CMSEException ) [pure virtual]`

Gets the name of this topic.

**Returns**

The topic name.

**Exceptions**

<b><i>CMSEException</i></b> (p. 1190)	- if an internal error occurs.
--	--------------------------------

Implemented in **activemq::commands::ActiveMQTempTopic** (p. 640).

The documentation for this class was generated from the following file:

- `src/main/cms/TemporaryTopic.h`

**6.808 cms::TextMessage Class Reference**

Interface for a text message.

```
#include <src/main/cms/TextMessage.h>
```

Inheritance diagram for `cms::TextMessage`:

**Public Member Functions**

- `virtual ~TextMessage ()`
- `virtual std::string getText () const =0 throw ( cms::CMSEException )`  
*Gets the message character buffer.*
- `virtual void setText (const char *msg)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )`  
*Sets the message contents, does not take ownership of the passed char\*, but copies it instead.*

- virtual void **setText** (const std::string &msg)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets the message contents.*

### 6.808.1 Detailed Description

Interface for a text message. A **TextMessage** (p. 3874) can contain any Text based payload such as an XML Document or other Text based document.

Like all Messages, a **TextMessage** (p. 3874) is received in Read-Only mode, any attempt to write to the **Message** (p. 2614) will result in a MessageNotWritableException being thrown until the `clearBody` method is called which will erase the contents and place the message back in a read / write mode.

#### Since

1.0

### 6.808.2 Constructor & Destructor Documentation

6.808.2.1 virtual cms::TextMessage::~TextMessage ( ) [inline, virtual]

### 6.808.3 Member Function Documentation

6.808.3.1 virtual std::string cms::TextMessage::getText ( ) const throw ( cms::CMSEException ) [pure virtual]

Gets the message character buffer.

#### Returns

The message character buffer.

#### Exceptions

<b>CMSEException</b> (p. 1190)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::commands::ActiveMQTextMessage** (p. 670).

6.808.3.2 virtual void cms::TextMessage::setText ( const std::string & msg ) throw ( cms::MessageNotWriteableException, cms::CMSEException ) [pure virtual]

Sets the message contents.

**Parameters**

<i>msg</i>	The message buffer.
------------	---------------------

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	- if an internal error occurs.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode..

Implemented in **activemq::commands::ActiveMQTextMessage** (p. 670).

**6.808.3.3** `virtual void cms::TextMessage::setText ( const char * msg ) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Sets the message contents, does not take ownership of the passed char\*, but copies it instead.

**Parameters**

<i>msg</i>	The message buffer.
------------	---------------------

**Exceptions**

<b><i>CMSException</i></b> (p. 1190)	- if an internal error occurs.
<b><i>MessageNotWriteableException</i></b> (p. 2808)	- if the message is in read-only mode..

Implemented in **activemq::commands::ActiveMQTextMessage** (p. 671).

The documentation for this class was generated from the following file:

- `src/main/cms/TextMessage.h`

## 6.809 decaf::lang::Thread Class Reference

A **Thread** (p. 3876) is a concurrent unit of execution.

```
#include <src/main/decaf/lang/Thread.h>
```

Inheritance diagram for decaf::lang::Thread:



## Data Structures

- class **UncaughtExceptionHandler**

*Interface for handlers invoked when a **Thread** (p. 3876) abruptly terminates due to an uncaught exception.*

## Public Types

- enum **State** {  
NEW = 0, RUNNABLE = 1, BLOCKED = 2, WAITING = 3,  
TIMED\_WAITING = 4, SLEEPING = 5, TERMINATED = 6 }

*Represents the various states that the **Thread** (p. 3876) can be in during its lifetime.*

## Public Member Functions

- **Thread** ()  
*Constructs a new **Thread** (p. 3876).*
- **Thread** (**Runnable** \*task)  
*Constructs a new **Thread** (p. 3876) with the given target **Runnable** (p. 3418) task.*
- **Thread** (const std::string &name)  
*Constructs a new **Thread** (p. 3876) with the given name.*
- **Thread** (**Runnable** \*task, const std::string &name)  
*Constructs a new **Thread** (p. 3876) with the given target **Runnable** (p. 3418) task and name.*
- virtual ~**Thread** ()
- virtual void **start** () throw ( decaf::lang::exceptions::IllegalThreadStateException, decaf::lang::exceptions::RuntimeException )  
*Creates a system thread and starts it in a joinable mode.*
- virtual void **join** () throw ( decaf::lang::exceptions::InterruptedException )  
*Forces the Current **Thread** (p. 3876) to wait until the thread exits.*
- virtual void **join** (long long millisecs) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::InterruptedException )  
*Forces the Current **Thread** (p. 3876) to wait until the thread exits.*
- virtual void **join** (long long millisecs, unsigned int nanos) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::InterruptedException )  
*Forces the Current **Thread** (p. 3876) to wait until the thread exits.*

- virtual void **run** ()  
*Default implementation of the run method - does nothing.*
- std::string **getName** () const  
*Returns the Thread's assigned name.*
- void **setName** (const std::string &name)  
*Sets the name of the **Thread** (p. 3876) to the new Name given by the argument name*
- int **getPriority** () const  
*Gets the currently set priority for this **Thread** (p. 3876).*
- void **setPriority** (int value) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Sets the current Thread's priority to the newly specified value.*
- const **UncaughtExceptionHandler** \* **getUncaughtExceptionHandler** () const  
*Set the handler invoked when this thread abruptly terminates due to an uncaught exception.*
- void **setUncaughtExceptionHandler** (**UncaughtExceptionHandler** \*handler)  
*Set the handler invoked when this thread abruptly terminates due to an uncaught exception.*
- std::string **toString** () const  
*Returns a string that describes the **Thread** (p. 3876).*
- bool **isAlive** () const  
*Returns true if the **Thread** (p. 3876) is alive, meaning it has been started and has not yet died.*
- **Thread::State** **getState** () const  
*Returns the currently set State of this **Thread** (p. 3876).*

### Static Public Member Functions

- static void **sleep** (long long millisecs) throw ( lang::exceptions::InterruptedException, lang::exceptions::IllegalArgumentException )  
*Causes the currently executing thread to halt execution for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers.*
- static void **sleep** (long long millisecs, unsigned int nanos) throw ( lang::exceptions::InterruptedException, lang::exceptions::IllegalArgumentException )

*Causes the currently executing thread to halt execution for the specified number of milliseconds plus any additionally specified nanoseconds given, subject to the precision and accuracy of system timers and schedulers.*

- static void **yield** ()

*Causes the currently executing thread object to temporarily pause and allow other threads to execute.*

- static long long **getId** ()

*Obtains the **Thread** (p. 3876) Id of the current thread.*

- static **Thread** \* **currentThread** ()

*Returns a pointer to the currently executing thread object.*

### Static Public Attributes

- static const int **MIN\_PRIORITY** = 1

*The minimum priority that a thread can have.*

- static const int **NORM\_PRIORITY** = 5

*The default priority that a thread is given at create time.*

- static const int **MAX\_PRIORITY** = 10

*The maximum priority that a thread can have.*

### Friends

- class **decaf::util::concurrent::locks::LockSupport**
- class **decaf::lang::Runtime**

#### 6.809.1 Detailed Description

A **Thread** (p. 3876) is a concurrent unit of execution. It has its own call stack for methods being invoked, their arguments and local variables. Each process has at least one main **Thread** (p. 3876) running when it is started; typically, there are several others for housekeeping. The application might decide to launch additional Threads for specific purposes.

Threads in the same process interact and synchronize by the use of shared objects and monitors associated with these objects.

There are basically two main ways of having a **Thread** (p. 3876) execute application code. One is providing a new class that extends **Thread** (p. 3876) and overriding its **run()** (p. 3884) method. The other is providing a new **Thread** (p. 3876) instance with a **Runnable** (p. 3418) object during its creation. In both cases, the **start()** (p. 3885) method must be called to actually execute the new **Thread** (p. 3876).

Each **Thread** (p. 3876) has an integer priority that basically determines the amount of CPU time the **Thread** (p. 3876) gets. It can be set using the **setPriority(int)** (p. 3884) method. A **Thread** (p. 3876) can also be made a daemon, which makes it run in the background. The latter also affects VM termination behavior: the VM does not terminate automatically as long as there are non-daemon threads running.

See also

**decaf.lang.ThreadGroup** (p. 3888)

Since

1.0

## 6.809.2 Member Enumeration Documentation

### 6.809.2.1 enum decaf::lang::Thread::State

Represents the various states that the **Thread** (p. 3876) can be in during its lifetime.

Enumerator:

**NEW** Before a **Thread** (p. 3876) is started it exists in this State.

**RUNNABLE** While a **Thread** (p. 3876) is running and is not blocked it is in this State.

**BLOCKED** A **Thread** (p. 3876) that is waiting to acquire a lock is in this state.

**WAITING** A **Thread** (p. 3876) that is waiting for another **Thread** (p. 3876) to perform an action is in this state.

**TIMED\_WAITING** A **Thread** (p. 3876) that is waiting for another **Thread** (p. 3876) to perform an action up to a specified time interval is in this state.

**SLEEPING** A **Thread** (p. 3876) that is blocked in a Sleep call is in this state.

**TERMINATED** A **Thread** (p. 3876) whose run method has exited is in this state.

## 6.809.3 Constructor & Destructor Documentation

### 6.809.3.1 decaf::lang::Thread::Thread ( )

Constructs a new **Thread** (p. 3876).

This constructor has the same effect as `Thread( NULL, NULL, GIVEN_NAME )`, where `GIVEN_NAME` is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

### 6.809.3.2 decaf::lang::Thread::Thread ( Runnable \* task )

Constructs a new **Thread** (p. 3876) with the given target **Runnable** (p. 3418) task.

This constructor has the same effect as `Thread( NULL, task, GIVEN_NAME )`, where `GIVEN_NAME` is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

#### Parameters

<i>task</i>	the <b>Runnable</b> (p. 3418) that this thread manages, if the task is <code>NULL</code> the Thread's run method is used instead.
-------------	---

#### 6.809.3.3 decaf::lang::Thread::Thread ( const std::string & name )

Constructs a new **Thread** (p. 3876) with the given name.

This constructor has the same effect as `Thread( NULL, NULL, GIVEN_NAME )`, where `GIVEN_NAME` is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

#### Parameters

<i>name</i>	the name to assign to this <b>Thread</b> (p. 3876).
-------------	---

#### 6.809.3.4 decaf::lang::Thread::Thread ( Runnable \* task, const std::string & name )

Constructs a new **Thread** (p. 3876) with the given target **Runnable** (p. 3418) task and name.

This constructor has the same effect as `Thread( NULL, task, GIVEN_NAME )`, where `GIVEN_NAME` is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

#### Parameters

<i>task</i>	the <b>Runnable</b> (p. 3418) that this thread manages, if the task is <code>NULL</code> the Thread's run method is used instead.
<i>name</i>	the name to assign to this <b>Thread</b> (p. 3876).

#### 6.809.3.5 virtual decaf::lang::Thread::~~Thread ( ) [virtual]

### 6.809.4 Member Function Documentation

#### 6.809.4.1 static Thread\* decaf::lang::Thread::currentThread ( ) [static]

Returns a pointer to the currently executing thread object.

#### Returns

**Pointer** (p. 3032) to the **Thread** (p. 3876) object representing the currently running **Thread** (p. 3876).

**6.809.4.2 static long long decaf::lang::Thread::getId ( ) [static]**

Obtains the **Thread** (p. 3876) Id of the current thread.

**Returns**

**Thread** (p. 3876) Id

**6.809.4.3 std::string decaf::lang::Thread::getName ( ) const**

Returns the **Thread**'s assigned name.

**Returns**

the Name of the **Thread** (p. 3876).

**6.809.4.4 int decaf::lang::Thread::getPriority ( ) const**

Gets the currently set priority for this **Thread** (p. 3876).

**Returns**

an int value representing the **Thread**'s current priority.

**6.809.4.5 Thread::State decaf::lang::Thread::getState ( ) const**

Returns the currently set State of this **Thread** (p. 3876).

**Returns**

the **Thread**'s current state.

**6.809.4.6 const UncaughtExceptionHandler\* decaf::lang::Thread::getUncaughtExceptionHandler ( ) const**

Set the handler invoked when this thread abruptly terminates due to an uncaught exception.

**Returns**

a pointer to the set **UncaughtExceptionHandler** (p. 4018).

**6.809.4.7** bool decaf::lang::Thread::isAlive ( ) const

Returns true if the **Thread** (p. 3876) is alive, meaning it has been started and has not yet died.

**Returns**

true if the thread is alive.

**6.809.4.8** virtual void decaf::lang::Thread::join ( long long *millisecs*, unsigned int *nanos* ) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::InterruptedException ) [virtual]

Forces the Current **Thread** (p. 3876) to wait until the thread exits.

**Parameters**

<i>millisecs</i>	the time in Milliseconds before the thread resumes
<i>nanos</i>	0-999999 extra nanoseconds to sleep.

**Exceptions**

<i>IllegalArgumentException</i>	if the nanoseconds parameter is out of range or the milliseconds paramter is negative.
<i>InterruptedException</i>	if any thread has interrupted the current thread. The interrupted status of the current thread is cleared when this exception is thrown.

**6.809.4.9** virtual void decaf::lang::Thread::join ( ) throw ( decaf::lang::exceptions::InterruptedException ) [virtual]

Forces the Current **Thread** (p. 3876) to wait until the thread exits.

**Exceptions**

<i>InterruptedException</i>	if any thread has interrupted the current thread. The interrupted status of the current thread is cleared when this exception is thrown.
-----------------------------	--

**6.809.4.10** virtual void decaf::lang::Thread::join ( long long *millisecs* ) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::InterruptedException ) [virtual]

Forces the Current **Thread** (p. 3876) to wait until the thread exits.

**Parameters**

<i>millisecs</i>	the time in Milliseconds before the thread resumes
------------------	--

**Exceptions**

<i>IllegalArgumentException</i>	if the milliseconds parameter is negative.
<i>InterruptedException</i>	if any thread has interrupted the current thread. The interrupted status of the current thread is cleared when this exception is thrown.

**6.809.4.11 virtual void decaf::lang::Thread::run ( ) [virtual]**

Default implementation of the run method - does nothing.

Implements **decaf::lang::Runnable** (p. 3419).

Reimplemented in **activemq::transport::mock::InternalCommandListener** (p. 2193), and **decaf::util::concurrent::PooledThread** (p. 3057).

**6.809.4.12 void decaf::lang::Thread::setName ( const std::string & name )**

Sets the name of the **Thread** (p. 3876) to the new Name given by the argument *name* name the new name of the **Thread** (p. 3876).

**6.809.4.13 void decaf::lang::Thread::setPriority ( int value ) throw ( decaf::lang::exceptions::IllegalArgumentException )**

Sets the current Thread's priority to the newly specified value.

The given value must be within the range **Thread::MIN\_PRIORITY** (p. 3886) and **Thread::MAX\_PRIORITY** (p. 3886).

**Parameters**

<i>value</i>	the new priority value to assign to this <b>Thread</b> (p. 3876).
--------------	---

**Exceptions**

<i>IllegalArgumentException</i>	if the value is out of range.
---------------------------------	-------------------------------

**6.809.4.14 void decaf::lang::Thread::setUncaughtExceptionHandler ( UncaughtExceptionHandler \* handler )**

Set the handler invoked when this thread abruptly terminates due to an uncaught exception.

**Parameters**

<i>handler</i>	the UncaughtExceptionHandler to invoke when the <b>Thread</b> (p. 3876) terminates due to an uncaught exception.
----------------	--



**6.809.4.15** static void decaf::lang::Thread::sleep ( long long *millisecs*, unsigned int *nanos* ) throw ( lang::exceptions::InterruptedException, lang::exceptions::IllegalArgumentException ) [static]

Causes the currently executing thread to halt execution for the specified number of milliseconds plus any additionally specified nanoseconds given, subject to the precision and accuracy of system timers and schedulers.

Note that this method is a static method that applies to the calling thread and not to the thread object.

#### Parameters

<i>millisecs</i>	time in milliseconds to halt execution.
<i>nanos</i>	0-999999 extra nanoseconds to sleep.

#### Exceptions

<i>IllegalArgumentException</i>	if the nanoseconds parameter is out of range or the milliseconds parameter is negative.
<i>InterruptedException</i>	if the <b>Thread</b> (p. 3876) was interrupted while sleeping.

**6.809.4.16** static void decaf::lang::Thread::sleep ( long long *millisecs* ) throw ( lang::exceptions::InterruptedException, lang::exceptions::IllegalArgumentException ) [static]

Causes the currently executing thread to halt execution for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers.

Note that this method is a static method that applies to the calling thread and not to the thread object.

#### Parameters

<i>millisecs</i>	time in milliseconds to halt execution.
------------------	---

#### Exceptions

<i>IllegalArgumentException</i>	if the milliseconds parameter is negative.
<i>InterruptedException</i>	if the <b>Thread</b> (p. 3876) was interrupted while sleeping.

**6.809.4.17** virtual void decaf::lang::Thread::start ( ) throw ( decaf::lang::exceptions::IllegalThreadStateException, decaf::lang::exceptions::RuntimeException ) [virtual]

Creates a system thread and starts it in a joinable mode.

Upon creation, the **run()** (p. 3884) method of either this object or the provided **Runnable** (p. 3418) object will be invoked in the context of this thread.

**Exceptions**

<i>IllegalThreadStateException</i>	if the thread has already been started.
<i>RuntimeException</i>	if the <b>Thread</b> (p. 3876) cannot be created for some reason.

**6.809.4.18** `std::string decaf::lang::Thread::toString ( ) const`

Returns a string that describes the **Thread** (p. 3876).

**Returns**

string describing the **Thread** (p. 3876).

**6.809.4.19** `static void decaf::lang::Thread::yield ( ) [static]`

Causes the currently executing thread object to temporarily pause and allow other threads to execute.

**6.809.5 Friends And Related Function Documentation****6.809.5.1** `friend class decaf::lang::Runtime [friend]`**6.809.5.2** `friend class decaf::util::concurrent::locks::LockSupport [friend]`**6.809.6 Field Documentation****6.809.6.1** `const int decaf::lang::Thread::MAX_PRIORITY = 10 [static]`

The maximum priority that a thread can have.

**6.809.6.2** `const int decaf::lang::Thread::MIN_PRIORITY = 1 [static]`

The minimum priority that a thread can have.

**6.809.6.3** `const int decaf::lang::Thread::NORM_PRIORITY = 5 [static]`

The default priority that a thread is given at create time.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Thread.h`

## 6.810 decaf::util::concurrent::ThreadFactory Class Reference

public interface **ThreadFactory** (p. 3887)

```
#include <src/main/decaf/util/concurrent/ThreadFactory.h>
```

### Public Member Functions

- virtual **~ThreadFactory** ()
- virtual **decaf::lang::Thread \* newThread** (decaf::lang::Runnable \*r)=0  
*Constructs a new Thread.*

#### 6.810.1 Detailed Description

public interface **ThreadFactory** (p. 3887) An object that creates new threads on demand. Using thread factories removes hardwiring of calls to new Thread, enabling applications to use special thread subclasses, priorities, etc.

The simplest implementation of this interface is just:

```
class SimpleThreadFactory : public ThreadFactory (p. 3887) { public: Thread* newThread(
Runnable* r ) { return new Thread(r); } }
```

The Executors.defaultThreadFactory() method provides a more useful simple implementation, that sets the created thread context to known values before returning it.

Since

1.0

#### 6.810.2 Constructor & Destructor Documentation

6.810.2.1 virtual decaf::util::concurrent::ThreadFactory::~~ThreadFactory ( ) [inline, virtual]

#### 6.810.3 Member Function Documentation

6.810.3.1 virtual decaf::lang::Thread\* decaf::util::concurrent::ThreadFactory::newThread ( decaf::lang::Runnable \* r ) [pure virtual]

Constructs a new Thread.

Implementations may also initialize priority, name, daemon status, ThreadGroup, etc. The pointer passed is still owned by the caller and is not deleted by the Thread object. The caller owns the returned Thread object and must delete it when finished.

#### Parameters

<i>r</i>	A pointer to a Runnable instance to be executed by new Thread instance returned.
----------	--

**Returns**

constructed thread, or NULL if the request to create a thread is rejected the caller owns the returned pointer.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ThreadFactory.h`

**6.811 decaf::lang::ThreadGroup Class Reference**

```
#include <src/main/decaf/lang/ThreadGroup.h>
```

**Public Member Functions**

- **ThreadGroup** ()
- virtual **~ThreadGroup** ()

**6.811.1 Detailed Description**

Since

1.0

**6.811.2 Constructor & Destructor Documentation**

**6.811.2.1 decaf::lang::ThreadGroup::ThreadGroup ( )**

**6.811.2.2 virtual decaf::lang::ThreadGroup::~~ThreadGroup ( ) [virtual]**

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/ThreadGroup.h`

**6.812 decaf::util::concurrent::ThreadPool Class Reference**

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.

```
#include <src/main/decaf/util/concurrent/ThreadPool.h>
```

Inheritance diagram for `decaf::util::concurrent::ThreadPool`:

## Public Types

- typedef std::pair< lang::Runnable \*, TaskListener \* > Task

## Public Member Functions

- **ThreadPool ()**
- virtual ~**ThreadPool ()**
- virtual void **queueTask** (Task task) throw ( lang::Exception )  
*Queue (p. 3239) a task to be completed by one of the Pooled Threads.*
- virtual **Task deQueueTask** () throw ( lang::Exception )  
*DeQueue a task to be completed by one of the Pooled Threads.*
- virtual std::size\_t **getPoolSize** () const  
*Returns the current number of Threads in the Pool, this is how many there are now, not how many are active or the max number that might exist.*
- virtual std::size\_t **getBacklog** () const  
*Returns the current backlog of items in the tasks queue, this is how much work is still waiting to get done.*
- virtual void **reserve** (std::size\_t size)  
*Ensures that there is at least the specified number of Threads allocated to the pool.*
- virtual std::size\_t **getMaxThreads** () const  
*Get the Max Number of Threads this Pool can contain.*
- virtual void **setMaxThreads** (std::size\_t maxThreads)  
*Sets the Max number of threads this pool can contain.*
- virtual std::size\_t **getBlockSize** () const  
*Gets the Max number of threads that can be allocated at a time when new threads are needed.*
- virtual void **setBlockSize** (std::size\_t blockSize)  
*Sets the Max number of Threads that can be allocated at a time when the Thread Pool determines that more Threads are needed.*
- virtual std::size\_t **getFreeThreadCount** () const  
*Returns the current number of available threads in the pool, threads that are performing a user task are considered unavailable.*
- virtual void **onTaskStarted** (PooledThread \*thread)  
*Called by a pooled thread when it is about to begin executing a new task.*
- virtual void **onTaskCompleted** (PooledThread \*thread)

*Called by a pooled thread when it has completed a task and is going back to waiting for another task to run, this will increment the free threads counter.*

- virtual void **onTaskException** (**PooledThread** \*thread, **lang::Exception** &ex)

*Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 3056) is now no longer running.*

### Static Public Member Functions

- static **ThreadPool** \* **getInstance** ()

*Return the one and only Thread Pool instance.*

### Static Public Attributes

- static const size\_t **DEFAULT\_MAX\_POOL\_SIZE** = 10
- static const size\_t **DEFAULT\_MAX\_BLOCK\_SIZE** = 3

#### 6.812.1 Detailed Description

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks. The Thread Pool has max size that it will grow to. The thread pool allocates threads in blocks. When there are no waiting worker threads and a task is queued then a new batch is allocated. The user can specify the size of the blocks, otherwise a default value is used.

When the user queues a task they must also queue a listener to be notified when the task has completed, this provides the user with a mechanism to know when a task object can be freed.

To have the Thread Pool perform a task, the user enqueue's an object that implements the `Runnable` interface and one of the worker threads will be executing it in its thread context.

## 6.812.2 Member Typedef Documentation

6.812.2.1 `typedef std::pair<lang::Runnable*, TaskListener*>`  
`decaf::util::concurrent::ThreadPool::Task`

## 6.812.3 Constructor & Destructor Documentation

6.812.3.1 `decaf::util::concurrent::ThreadPool::ThreadPool ( )`

6.812.3.2 `virtual decaf::util::concurrent::ThreadPool::~~ThreadPool ( )` `[virtual]`

## 6.812.4 Member Function Documentation

6.812.4.1 `virtual Task decaf::util::concurrent::ThreadPool::deQueueTask ( )` `throw ( lang::Exception )` `[virtual]`

DeQueue a task to be completed by one of the Pooled Threads.

A caller of this method will block until there is something in the tasks queue, therefore care must be taken when calling this function. Normally clients of **ThreadPool** (p. 3888) don't use this, only the **PooledThread** (p. 3056) objects owned by this **ThreadPool** (p. 3888).

### Returns

object that derives from Runnable

### Exceptions

<i>ActiveMQException</i>	
--------------------------	--

6.812.4.2 `virtual std::size_t decaf::util::concurrent::ThreadPool::getBacklog ( )` `const`  
`[inline, virtual]`

Returns the current backlog of items in the tasks queue, this is how much work is still waiting to get done.

### Returns

number of outstanding tasks.

6.812.4.3 `virtual std::size_t decaf::util::concurrent::ThreadPool::getBlockSize ( )` `const`  
`[inline, virtual]`

Gets the Max number of threads that can be allocated at a time when new threads are needed.

### Returns

max Thread Block Size

**6.812.4.4** `virtual std::size_t decaf::util::concurrent::ThreadPool::getFreeThreadCount ( ) const`  
[inline, virtual]

Returns the current number of available threads in the pool, threads that are performing a user task are considered unavailable.

This value could change immediately after calling as Threads could finish right after and be available again. This is informational only.

#### Returns

total free threads

**6.812.4.5** `static ThreadPool* decaf::util::concurrent::ThreadPool::getInstance ( )`  
[static]

Return the one and only Thread Pool instance.

#### Returns

The Thread Pool Pointer

**6.812.4.6** `virtual std::size_t decaf::util::concurrent::ThreadPool::getMaxThreads ( ) const`  
[inline, virtual]

Get the Max Number of Threads this Pool can contain.

#### Returns

max size

**6.812.4.7** `virtual std::size_t decaf::util::concurrent::ThreadPool::getPoolSize ( ) const`  
[inline, virtual]

Returns the current number of Threads in the Pool, this is how many there are now, not how many are active or the max number that might exist.

#### Returns

integer number of threads in existence.

**6.812.4.8** `virtual void decaf::util::concurrent::ThreadPool::onTaskCompleted ( PooledThread * thread )` [virtual]

Called by a pooled thread when it has completed a task and is going back to waiting for another task to run, this will increment the free threads counter.

#### Parameters



<i>thread</i>	Pointer the the Pooled Thread that is making this call.
---------------	---

Implements **decaf::util::concurrent::PooledThreadListener** (p. 3059).

**6.812.4.9** `virtual void decaf::util::concurrent::ThreadPool::onTaskException ( PooledThread * thread, lang::Exception & ex )` [virtual]

Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 3056) is now no longer running.

#### Parameters

<i>thread</i>	Pointer to the Pooled Thread that is making this call
<i>ex</i>	The Exception that occurred.

Implements **decaf::util::concurrent::PooledThreadListener** (p. 3059).

**6.812.4.10** `virtual void decaf::util::concurrent::ThreadPool::onTaskStarted ( PooledThread * thread )` [virtual]

Called by a pooled thread when it is about to begin executing a new task.

This will decrement the available threads counter so that this object knows when there are no more free threads and must create new ones.

#### Parameters

<i>thread</i>	Pointer to the Pooled Thread that is making this call
---------------	---

Implements **decaf::util::concurrent::PooledThreadListener** (p. 3059).

**6.812.4.11** `virtual void decaf::util::concurrent::ThreadPool::queueTask ( Task task ) throw ( lang::Exception )` [virtual]

**Queue** (p. 3239) a task to be completed by one of the Pooled Threads.

tasks are serviced as soon as a **PooledThread** (p. 3056) is available to run it.

#### Parameters

<i>task</i>	object that derives from Runnable
-------------	-----------------------------------

#### Exceptions

<i>ActiveMQException</i>	
--------------------------	--

**6.812.4.12** `virtual void decaf::util::concurrent::ThreadPool::reserve ( std::size_t size )`  
`[virtual]`

Ensures that there is at least the specified number of Threads allocated to the pool.

If the size is greater than the MAX number of threads in the pool, then only MAX threads are reservved. If the size is smaller than the number of threads currently in the pool, than nothing is done.

#### Parameters

<i>size</i>	the number of threads to reserve.
-------------	-----------------------------------

**6.812.4.13** `virtual void decaf::util::concurrent::ThreadPool::setBlockSize ( std::size_t blockSize )`  
`[virtual]`

Sets the Max number of Threads that can be allocated at a time when the Thread Pool determines that more Threads are needed.

#### Parameters

<i>blockSize</i>	Max Thread Block Size
------------------	-----------------------

**6.812.4.14** `virtual void decaf::util::concurrent::ThreadPool::setMaxThreads ( std::size_t maxThreads )`  
`[virtual]`

Sets the Max number of threads this pool can contian.

if this value is smaller than the current size of the pool nothing is done.

#### Parameters

<i>maxThreads</i>	total number of threads that can be pooled
-------------------	--

### 6.812.5 Field Documentation

**6.812.5.1** `const size_t decaf::util::concurrent::ThreadPool::DEFAULT_MAX_BLOCK_SIZE = 3` `[static]`

**6.812.5.2** `const size_t decaf::util::concurrent::ThreadPool::DEFAULT_MAX_POOL_SIZE = 10` `[static]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ThreadPool.h`

## 6.813 decaf::lang::Throwable Class Reference

This class represents an error that has occurred.

```
#include <src/main/decaf/lang/Throwable.h>
```

Inheritance diagram for decaf::lang::Throwable:

### Public Member Functions

- **Throwable** () throw ()
- virtual ~**Throwable** () throw ()
- virtual std::string **getMessage** () const =0  
*Gets the cause of the error; if no message was provided to the instance of this interface but a cause was then the value cause.getMessage is then returned.*
- virtual const std::exception \* **getCause** () const =0  
*Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.*
- virtual void **initCause** (const std::exception \*cause)=0  
*Initializes the contained cause exception with the one given.*
- virtual void **setMark** (const char \*file, const int lineNumber)=0  
*Adds a file/line number to the stack trace.*
- virtual **Throwable** \* **clone** () const =0  
*Clones this exception.*
- virtual std::vector< std::pair< std::string, int > > **getStackTrace** () const =0  
*Provides the stack trace for every point where this exception was caught, marked, and rethrown.*
- virtual void **printStackTrace** () const =0  
*Prints the stack trace to std::err.*
- virtual void **printStackTrace** (std::ostream &stream) const =0  
*Prints the stack trace to the given output stream.*
- virtual std::string **getStackTraceString** () const =0  
*Gets the stack trace as one contiguous string.*

### 6.813.1 Detailed Description

This class represents an error that has occurred. All Exceptions in the Decaf library should extend from this or from the **Exception** (p. 1886) class in order to ensure that all Decaf Exceptions are interchangeable with the `std::exception` class.

**Throwable** (p. 3895) can wrap another **Throwable** (p. 3895) as the cause if the error being thrown. The user can inspect the cause by calling `getCause`, the pointer returned is the property of the **Throwable** (p. 3895) instance and will be deleted when it is deleted or goes out of scope.

Since

1.0

### 6.813.2 Constructor & Destructor Documentation

6.813.2.1 `decaf::lang::Throwable::Throwable ( ) throw ()` [inline]

6.813.2.2 `virtual decaf::lang::Throwable::~~Throwable ( ) throw ()` [inline, virtual]

### 6.813.3 Member Function Documentation

6.813.3.1 `virtual Throwable* decaf::lang::Throwable::clone ( ) const` [pure virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

Copy of this **Exception** (p. 1886) object

Implemented in `activemq::exceptions::ActiveMQException` (p. 349), `activemq::exceptions::BrokerException` (p. 874), `decaf::internal::net::ssl::openssl::OpenSSLSocketException` (p. 2958), `decaf::io::EOFException` (p. 1883), `decaf::io::InterruptedIOException` (p. 2198), `decaf::io::IOException` (p. 2212), `decaf::io::UnsupportedEncodingException` (p. 4027), `decaf::io::UTFDataFormatException` (p. 4080), `decaf::lang::Exception` (p. 1889), `decaf::lang::exceptions::ClassCastException` (p. 1179), `decaf::lang::exceptions::IllegalArgumentExceptio` (p. 2056), `decaf::lang::exceptions::IllegalMonitorStateException` (p. 2059), `decaf::lang::exceptions::Illegal` (p. 2062), `decaf::lang::exceptions::IllegalThreadStateException` (p. 2065), `decaf::lang::exceptions::IndexC` (p. 2072), `decaf::lang::exceptions::InterruptedException` (p. 2196), `decaf::lang::exceptions::InvalidStateE` (p. 2210), `decaf::lang::exceptions::NoSuchElementException` (p. 2912), `decaf::lang::exceptions::NullPoint` (p. 2918), `decaf::lang::exceptions::NumberFormatException` (p. 2923), `decaf::lang::exceptions::RuntimeE` (p. 3423), `decaf::lang::exceptions::UnsupportedOperationException` (p. 4030), `decaf::net::BindException` (p. 844), `decaf::net::ConnectException` (p. 1296), `decaf::net::HttpRetryException` (p. 2051), `decaf::net::MalformedURLException` (p. 2538), `decaf::net::NoRouteToHostException`

(p. 2907), **decaf::net::PortUnreachableException** (p. 3062), **decaf::net::ProtocolException** (p. 3230), **decaf::net::SocketException** (p. 3629), **decaf::net::SocketTimeoutException** (p. 3652), **decaf::net::UnknownHostException** (p. 4022), **decaf::net::UnknownServiceException** (p. 4024), **decaf::net::URISyntaxException** (p. 4063), **decaf::nio::BufferOverflowException** (p. 966), **decaf::nio::BufferUnderflowException** (p. 969), **decaf::nio::InvalidMarkException** (p. 2206), **decaf::nio::ReadOnlyBufferException** (p. 3263), **decaf::security::cert::CertificateEncodingException** (p. 1118), **decaf::security::cert::CertificateException** (p. 1120), **decaf::security::cert::CertificateExpiredException** (p. 1122), **decaf::security::cert::CertificateNotYetValidException** (p. 1124), **decaf::security::cert::CertificateParsingException** (p. 1126), **decaf::security::GeneralSecurityException** (p. 2037), **decaf::security::InvalidKeyException** (p. 2203), **decaf::security::KeyException** (p. 2368), **decaf::security::KeyManagementException** (p. 2371), **decaf::security::NoSuchAlgorithmException** (p. 2910), **decaf::security::NoSuchProviderException** (p. 2915), **decaf::security::SignatureException** (p. 3604), **decaf::util::concurrent::BrokenBarrierException** (p. 868), **decaf::util::concurrent::CancellationException** (p. 1112), **decaf::util::concurrent::ExecutionException** (p. 1926), **decaf::util::concurrent::RejectedExecutionException** (p. 3282), **decaf::util::concurrent::TimeoutException** (p. 3901), **decaf::util::zip::DataFormatException** (p. 1602), and **decaf::util::zip::ZipException** (p. 4178).

**6.813.3.2** `virtual const std::exception* decaf::lang::Throwable::getCause ( ) const [pure virtual]`

Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

#### Returns

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

Implemented in **decaf::lang::Exception** (p. 1890).

**6.813.3.3** `virtual std::string decaf::lang::Throwable::getMessage ( ) const [pure virtual]`

Gets the cause of the error, if no message was provided to the instance of this interface but a cause was then the value `cause.getMessage` is then returned.

#### Returns

string errors message

Implemented in **decaf::lang::Exception** (p. 1891).

**6.813.3.4** `virtual std::vector< std::pair< std::string, int> > decaf::lang::Throwable::getStackTrace ( ) const [pure virtual]`

Provides the stack trace for every point where this exception was caught, marked, and rethrown.

**Returns**

vector containing stack trace strings

Implemented in **decaf::lang::Exception** (p. 1891).

**6.813.3.5** `virtual std::string decaf::lang::Throwable::getStackTraceString ( ) const` [pure virtual]

Gets the stack trace as one contiguous string.

**Returns**

string with formatted stack trace data

Implemented in **decaf::lang::Exception** (p. 1891).

**6.813.3.6** `virtual void decaf::lang::Throwable::initCause ( const std::exception * cause )` [pure virtual]

Initializes the contained cause exception with the one given.

A copy is made to avoid ownership issues.

**Parameters**

<i>cause</i>	The exception that was the cause of this one.
--------------	---

Implemented in **decaf::lang::Exception** (p. 1891).

**6.813.3.7** `virtual void decaf::lang::Throwable::printStackTrace ( ) const` [pure virtual]

Prints the stack trace to std::err.

Implemented in **decaf::lang::Exception** (p. 1892).

**6.813.3.8** `virtual void decaf::lang::Throwable::printStackTrace ( std::ostream & stream ) const` [pure virtual]

Prints the stack trace to the given output stream.

**Parameters**

<i>stream</i>	the target output stream.
---------------	---------------------------

Implemented in **decaf::lang::Exception** (p. 1892).

6.813.3.9 virtual void decaf::lang::Throwable::setMark ( const char \* *file*, const int *lineNumber* ) [pure virtual]

Adds a file/line number to the stack trace.

#### Parameters

<i>file</i>	The name of the file calling this method (use <code>__FILE__</code> ).
<i>lineNumber</i>	The line number in the calling file (use <code>__LINE__</code> ).

Implemented in **decaf::lang::Exception** (p. 1892).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Throwable.h**

## 6.814 decaf::util::concurrent::TimeoutException Class Reference

```
#include <src/main/decaf/util/concurrent/TimeoutException.h>
```

Inheritance diagram for decaf::util::concurrent::TimeoutException:

### Public Member Functions

- **TimeoutException** () throw ()  
*Default Constructor.*
- **TimeoutException** (const **decaf::lang::Exception** &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **TimeoutException** (const **TimeoutException** &ex) throw ()  
*Copy Constructor.*
- **TimeoutException** (const std::exception \***cause**) throw ()  
*Constructor.*
- **TimeoutException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **TimeoutException** (const char \*file, const int lineNumber, const std::exception \***cause**, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **TimeoutException** \* **clone** () const

*Clones this exception.*

- virtual `~TimeoutException () throw ()`

### 6.814.1 Constructor & Destructor Documentation

**6.814.1.1** `decaf::util::concurrent::TimeoutException::TimeoutException ( ) throw ()`  
[inline]

Default Constructor.

**6.814.1.2** `decaf::util::concurrent::TimeoutException::TimeoutException ( const decaf::lang::Exception & ex ) throw ()` [inline]

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.814.1.3** `decaf::util::concurrent::TimeoutException::TimeoutException ( const TimeoutException & ex ) throw ()` [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	The exception to copy from.
-----------	-----------------------------

**6.814.1.4** `decaf::util::concurrent::TimeoutException::TimeoutException ( const std::exception * cause ) throw ()` [inline]

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.814.1.5** `decaf::util::concurrent::TimeoutException::TimeoutException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message



**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The string message to report
...	list of primitives that are formatted into the message

**6.814.1.6** `decaf::util::concurrent::TimeoutException::TimeoutException ( const char * file,  
const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()`  
[inline]

Constructor - Initializes the file name and line number where this message occurred.  
Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The string message to report
...	list of primitives that are formatted into the message

**6.814.1.7** `virtual decaf::util::concurrent::TimeoutException::~~TimeoutException ( ) throw ()`  
[inline, virtual]

**6.814.2 Member Function Documentation**

**6.814.2.1** `virtual TimeoutException* decaf::util::concurrent::TimeoutException::clone ( )`  
`const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns**

a new **TimeoutException** (p. 3899) that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1889).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/TimeoutException.h`

**6.815 decaf::util::Timer Class Reference**

A facility for threads to schedule tasks for future execution in a background thread.

```
#include <src/main/decaf/util/Timer.h>
```

## Public Member Functions

- **Timer** ()
- virtual  $\sim$ **Timer** ()
- void **cancel** ()

*Terminates this timer, discarding any currently scheduled tasks.*

- std::size\_t **purge** ()

*Removes all canceled tasks from this timer's task queue.*

- void **schedule** (**TimerTask** \*task, long long delay) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Schedules the specified task for execution after the specified delay.*

- void **schedule** (const decaf::lang::Pointer< **TimerTask** > &task, long long delay) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Schedules the specified task for execution after the specified delay.*

- void **schedule** (**TimerTask** \*task, const **Date** &time) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Schedules the specified task for execution at the specified time.*

- void **schedule** (const decaf::lang::Pointer< **TimerTask** > &task, const **Date** &time) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Schedules the specified task for execution at the specified time.*

- void **schedule** (**TimerTask** \*task, long long delay, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.*

- void **schedule** (const decaf::lang::Pointer< **TimerTask** > &task, long long delay, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.*

- void **schedule** (**TimerTask** \*task, const **Date** &firstTime, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.*

- void **schedule** (const **decaf::lang::Pointer**< **TimerTask** > &task, const **Date** &firstTime, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.*

- void **scheduleAtFixedRate** (**TimerTask** \*task, long long delay, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.*

- void **scheduleAtFixedRate** (const **decaf::lang::Pointer**< **TimerTask** > &task, long long delay, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.*

- void **scheduleAtFixedRate** (**TimerTask** \*task, const **Date** &firstTime, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.*

- void **scheduleAtFixedRate** (const **decaf::lang::Pointer**< **TimerTask** > &task, const **Date** &firstTime, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.*

### 6.815.1 Detailed Description

A facility for threads to schedule tasks for future execution in a background thread. Tasks may be scheduled for one-time execution, or for repeated execution at regular intervals.

Corresponding to each **Timer** (p. 3901) object is a single background thread that is used to execute all of the timer's tasks, sequentially. **Timer** (p. 3901) tasks should complete quickly. If a timer task takes excessive time to complete, it "hogs" the timer's task execution thread. This can, in turn, delay the execution of subsequent tasks, which may "bunch up" and execute in rapid succession when (and if) the offending task finally completes.

This class is thread-safe: multiple threads can share a single **Timer** (p. 3901) object without the need for external synchronization.

This class does not offer real-time guarantees: it schedules tasks using the `wait(long)` method.

#### Since

1.0

### 6.815.2 Constructor & Destructor Documentation

6.815.2.1 `decaf::util::Timer::Timer ( )`

6.815.2.2 `virtual decaf::util::Timer::~~Timer ( )` [virtual]

### 6.815.3 Member Function Documentation

6.815.3.1 `void decaf::util::Timer::cancel ( )`

Terminates this timer, discarding any currently scheduled tasks.

Does not interfere with a currently executing task (if it exists). Once a timer has been terminated, its execution thread terminates gracefully, and no more tasks may be scheduled on it.

Note that calling this method from within the `run` method of a timer task that was invoked by this timer absolutely guarantees that the ongoing task execution is the last task execution that will ever be performed by this timer.

This method may be called repeatedly; the second and subsequent calls have no effect.

6.815.3.2 `std::size_t decaf::util::Timer::purge ( )`

Removes all canceled tasks from this timer's task queue.

Calling this method has no effect on the behavior of the timer, but eliminates the canceled tasks from the queue causing the **Timer** (p. 3901) to destroy the **TimerTask** (p. 3914) pointer it was originally given, the caller should ensure that they no longer have any references to **TimerTasks** that were previously scheduled.

Most programs will have no need to call this method. It is designed for use by the rare application that cancels a large number of tasks. Calling this method trades time for space: the runtime of the method may be proportional to  $n + c \log n$ , where  $n$  is the number of tasks in the queue and  $c$  is the number of canceled tasks.

This method can be called on a **Timer** (p. 3901) object that has no scheduled tasks without error.

#### Returns

the number of tasks removed from the queue.

```
6.815.3.3 void decaf::util::Timer::schedule ( const decaf::lang::Pointer<
    TimerTask > & task, long long delay ) throw (
    decaf::lang::exceptions::NullPointerException,
    decaf::lang::exceptions::IllegalArgumentException,
    decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for execution after the specified delay.

#### Parameters

<i>task</i>	- task to be scheduled.
<i>delay</i>	- delay in milliseconds before task is to be executed.

#### Exceptions

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3914) value is Null.
<i>IllegalArgumentException</i>	- if delay is negative, or delay + System.currentTimeMillis() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, or timer was cancelled.

```
6.815.3.4 void decaf::util::Timer::schedule ( const decaf::lang::Pointer<
    TimerTask > & task, long long delay, long long period )
    throw ( decaf::lang::exceptions::NullPointerException,
    decaf::lang::exceptions::IllegalArgumentException,
    decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.

Subsequent executions take place at approximately regular intervals separated by the specified period.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying Object.wait(long long) is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

#### Parameters

<i>task</i>	- task to be scheduled.
<i>delay</i>	- delay in milliseconds before task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

**Exceptions**

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3914) value is Null.
<i>IllegalArgumentException</i>	- if delay is negative, or delay + System.currentTimeMillis() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

**6.815.3.5** `void decaf::util::Timer::schedule ( TimerTask * task, const Date & firstTime, long long period ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )`

Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.

Subsequent executions take place at approximately regular intervals separated by the specified period.

The **TimerTask** (p. 3914) pointer is considered to be owned by the **Timer** (p. 3901) class once it has been scheduled, the **Timer** (p. 3901) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3901) itself is cancelled. A **TimerTask** (p. 3914) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3901) and the caller should ensure that the **TimerTask** (p. 3914) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3914) instance are planned.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying Object.wait(long long) is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

**Parameters**

<i>task</i>	- task to be scheduled.
<i>firstTime</i>	- First time at which task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

**Exceptions**

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3914) value is Null.
<i>IllegalArgumentException</i>	- if time.getTime() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

**6.815.3.6** void decaf::util::Timer::schedule ( **TimerTask** \* *task*, const Date & *time* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

Schedules the specified task for execution at the specified time.

If the time is in the past, the task is scheduled for immediate execution.

The **TimerTask** (p. 3914) pointer is considered to be owned by the **Timer** (p. 3901) class once it has been scheduled, the **Timer** (p. 3901) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3901) itself is cancelled. A **TimerTask** (p. 3914) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3901) and the caller should ensure that the **TimerTask** (p. 3914) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3914) instance are planned.

#### Parameters

<i>task</i>	- task to be scheduled.
<i>time</i>	- time at which task is to be executed.

#### Exceptions

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3914) value is Null.
<i>IllegalArgumentException</i>	- if time.getTime() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

**6.815.3.7** void decaf::util::Timer::schedule ( const decaf::lang::Pointer< **TimerTask** > & *task*, const Date & *firstTime*, long long *period* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.

Subsequent executions take place at approximately regular intervals separated by the specified period.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long long)` is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

#### Parameters

<i>task</i>	- task to be scheduled.
<i>firstTime</i>	- First time at which task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

#### Exceptions

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3914) value is Null.
<i>IllegalArgumentException</i>	- if <code>time.getTime()</code> is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

**6.815.3.8** `void decaf::util::Timer::schedule ( TimerTask * task, long long delay ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )`

Schedules the specified task for execution after the specified delay.

The **TimerTask** (p. 3914) pointer is considered to be owned by the **Timer** (p. 3901) class once it has been scheduled, the **Timer** (p. 3901) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3901) itself is cancelled. A **TimerTask** (p. 3914) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3901) and the caller should ensure that the **TimerTask** (p. 3914) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3914) instance are planned.

#### Parameters

<i>task</i>	- task to be scheduled.
<i>delay</i>	- delay in milliseconds before task is to be executed.



**Exceptions**

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3914) value is Null.
<i>IllegalArgumentException</i>	- if delay is negative, or delay + System.currentTimeMillis() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, or timer was cancelled.

**6.815.3.9** void decaf::util::Timer::schedule ( **TimerTask** \* *task*, long long *delay*, long long *period* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.

Subsequent executions take place at approximately regular intervals separated by the specified period.

The **TimerTask** (p. 3914) pointer is considered to be owned by the **Timer** (p. 3901) class once it has been scheduled, the **Timer** (p. 3901) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3901) itself is cancelled. A **TimerTask** (p. 3914) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3901) and the caller should ensure that the **TimerTask** (p. 3914) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3914) instance are planned.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying Object.wait(long long) is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

**Parameters**

<i>task</i>	- task to be scheduled.
<i>delay</i>	- delay in milliseconds before task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

**Exceptions**

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3914) value is Null.
-----------------------------	--

<i>IllegalArgumentException</i>	- if delay is negative, or delay + System.currentTimeMillis() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

**6.815.3.10** `void decaf::util::Timer::schedule ( const decaf::lang::Pointer< TimerTask > & task, const Date & time ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )`

Schedules the specified task for execution at the specified time.

If the time is in the past, the task is scheduled for immediate execution.

#### Parameters

<i>task</i>	- task to be scheduled.
<i>time</i>	- time at which task is to be executed.

#### Exceptions

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3914) value is Null.
<i>IllegalArgumentException</i>	- if time.getTime() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

**6.815.3.11** `void decaf::util::Timer::scheduleAtFixedRate ( TimerTask * task, long long delay, long long period ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )`

Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.

Subsequent executions take place at approximately regular intervals, separated by the specified period.

The **TimerTask** (p. 3914) pointer is considered to be owned by the **Timer** (p. 3901) class once it has been scheduled, the **Timer** (p. 3901) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3901) itself is cancelled. A **TimerTask** (p. 3914) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3901) and the caller should ensure that the **TimerTask** (p. 3914) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3914) instance are planned.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long)` is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

### Parameters

<i>task</i>	- task to be scheduled.
<i>delay</i>	- delay in milliseconds before task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

### Exceptions

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3914) value is Null.
<i>IllegalArgumentException</i>	- if delay is negative, or delay + <code>System.currentTimeMillis()</code> is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

```
6.815.3.12 void decaf::util::Timer::scheduleAtFixedRate ( const decaf::lang::Pointer<
TimerTask > & task, long long delay, long long period )
throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.

Subsequent executions take place at approximately regular intervals, separated by the specified period.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long)` is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance

every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

#### Parameters

<i>task</i>	- task to be scheduled.
<i>delay</i>	- delay in milliseconds before task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

#### Exceptions

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3914) value is Null.
<i>IllegalArgumentException</i>	- if delay is negative, or delay + System.currentTimeMillis() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

**6.815.3.13** `void decaf::util::Timer::scheduleAtFixedRate ( const decaf::lang::Pointer< TimerTask > & task, const Date & firstTime, long long period ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )`

Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.

Subsequent executions take place at approximately regular intervals, separated by the specified period.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying Object.wait(long) is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

#### Parameters

<i>task</i>	- task to be scheduled.
-------------	-------------------------

<i>firstTime</i>	- First time at which task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

### Exceptions

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3914) value is Null.
<i>IllegalArgumentException</i>	- if time.getTime() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

**6.815.3.14** void decaf::util::Timer::scheduleAtFixedRate ( **TimerTask**  
 \* *task*, const Date & *firstTime*, long long *period* ) throw  
 ( decaf::lang::exceptions::NullPointerException,  
 decaf::lang::exceptions::IllegalArgumentException,  
 decaf::lang::exceptions::IllegalStateException )

Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.

Subsequent executions take place at approximately regular intervals, separated by the specified period.

The **TimerTask** (p. 3914) pointer is considered to be owned by the **Timer** (p. 3901) class once it has been scheduled, the **Timer** (p. 3901) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3901) itself is cancelled. A **TimerTask** (p. 3914) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3901) and the caller should ensure that the **TimerTask** (p. 3914) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3914) instance are planned.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying Object.wait(long) is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

### Parameters

<i>task</i>	- task to be scheduled.
<i>firstTime</i>	- First time at which task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

### Exceptions

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3914) value is Null.
<i>IllegalArgumentException</i>	- if time.getTime() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Timer.h**

## 6.816 decaf::util::TimerTask Class Reference

A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 3901).

```
#include <src/main/decaf/util/TimerTask.h>
```

Inheritance diagram for decaf::util::TimerTask:

### Public Member Functions

- **TimerTask** ()
- virtual **~TimerTask** ()
- bool **cancel** ()  
*Cancels this timer task.*
- long long **scheduledExecutionTime** () const  
*Returns the scheduled execution time of the most recent actual execution of this task.*

### Protected Member Functions

- bool **isScheduled** () const
- void **setScheduledTime** (long long time)
- long long **getWhen** () const

## Friends

- class **Timer**
- class **TimerImpl**
- class **decaf::internal::util::TimerTaskHeap**

## 6.816.1 Detailed Description

A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 3901).

### Since

1.0

## 6.816.2 Constructor & Destructor Documentation

**6.816.2.1** `decaf::util::TimerTask::TimerTask ( )`

**6.816.2.2** `virtual decaf::util::TimerTask::~~TimerTask ( )` [`inline`, `virtual`]

## 6.816.3 Member Function Documentation

**6.816.3.1** `bool decaf::util::TimerTask::cancel ( )`

Cancels this timer task.

If the task has been scheduled for one-time execution and has not yet run, or has not yet been scheduled, it will never run. If the task has been scheduled for repeated execution, it will never run again. (If the task is running when this call occurs, the task will run to completion, but will never run again.)

Note that calling this method from within the run method of a repeating timer task absolutely guarantees that the timer task will not run again.

This method may be called repeatedly; the second and subsequent calls have no effect.

## Returns

true if this task is scheduled for one-time execution and has not yet run, or this task is scheduled for repeated execution. Returns false if the task was scheduled for one-time execution and has already run, or if the task was never scheduled, or if the task was already canceled. (Loosely speaking, this method returns true if it prevents one or more scheduled executions from taking place.)

**6.816.3.2** `long long decaf::util::TimerTask::getWhen ( ) const` [protected]

**6.816.3.3** `bool decaf::util::TimerTask::isScheduled ( ) const` [protected]

**6.816.3.4** `long long decaf::util::TimerTask::scheduledExecutionTime ( ) const`

Returns the scheduled execution time of the most recent actual execution of this task.

(If this method is invoked while task execution is in progress, the return value is the scheduled execution time of the ongoing task execution.)

This method is typically invoked from within a task's run method, to determine whether the current execution of the task is sufficiently timely to warrant performing the scheduled activity:

```
void run() (p. 3419) { if( System::currentTimeMillis() - scheduledExecutionTime()
(p. 3916) >= MAX_TARDINESS) return; // Too late; skip this execution. // Perform
the task }
```

This method is typically not used in conjunction with fixed-delay execution repeating tasks, as their scheduled execution times are allowed to drift over time, and so are not terribly significant.

### Returns

the time at which the most recent execution of this task was scheduled to occur, in the format returned by **Date.getTime()** (p. 1722). The return value is undefined if the task has yet to commence its first execution.

**6.816.3.5** `void decaf::util::TimerTask::setScheduledTime ( long long time )` [protected]

## 6.816.4 Friends And Related Function Documentation

**6.816.4.1** `friend class decaf::internal::util::TimerTaskHeap` [friend]

**6.816.4.2** `friend class Timer` [friend]

**6.816.4.3** `friend class TimerImpl` [friend]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/TimerTask.h`

## 6.817 decaf::internal::util::TimerTaskHeap Class Reference

A Binary Heap implemented specifically for the Timer class in Decaf Util.

```
#include <src/main/decaf/internal/util/TimerTaskHeap.h>
```



## Public Member Functions

- **TimerTaskHeap** ()
- virtual **~TimerTaskHeap** ()
- **Pointer**< **TimerTask** > **peek** ()

*Peeks at the Head of the Heap, returns the task with the nearest scheduled run time.*

- bool **isEmpty** () const
- std::size\_t **size** () const
- void **insert** (const **Pointer**< **TimerTask** > &task)

*Inserts the specified Task into the heap, heap is reordered to reflect the addition of a new element.*

- void **remove** (std::size\_t pos)

*Removes the Task at the specified position from the heap, resorts the heap from the position down to the bottom.*

- void **reset** ()

*Clear all contents from the heap.*

- void **adjustMinimum** ()

*Resorts the heap starting at the top.*

- std::size\_t **deleteIfCancelled** ()

*Runs through the heap removing all cancelled Tasks from it, this is not normally used but in case a cancellation of a large number of tasks the user can perform this purge.*

- std::size\_t **find** (const **Pointer**< **TimerTask** > &task) const

*Searches the heap for the specified TimerTask element and returns its position in the heap.*

### 6.817.1 Detailed Description

A Binary Heap implemented specifically for the Timer class in Decaf Util.

Since

1.0

## 6.817.2 Constructor & Destructor Documentation

6.817.2.1 `decaf::internal::util::TimerTaskHeap::TimerTaskHeap ( )`

6.817.2.2 `virtual decaf::internal::util::TimerTaskHeap::~~TimerTaskHeap ( )` [*virtual*]

## 6.817.3 Member Function Documentation

6.817.3.1 `void decaf::internal::util::TimerTaskHeap::adjustMinimum ( )`

Resorts the heap starting at the top.

6.817.3.2 `std::size_t decaf::internal::util::TimerTaskHeap::deleteIfCancelled ( )`

Runs through the heap removing all cancelled Tasks from it, this is not normally used but in case a a cancellation of a large number of tasks the user can perform this purge.

### Returns

the number of task that were removed from the heap because they were cancelled.

6.817.3.3 `std::size_t decaf::internal::util::TimerTaskHeap::find ( const Pointer< TimerTask > & task ) const`

Searches the heap for the specified TimerTask element and returns its position in the heap.

Returns the unsigned equivalent of -1 if the element is not found.

### Returns

the position in the Heap where the Task is stored, or npos.

6.817.3.4 `void decaf::internal::util::TimerTaskHeap::insert ( const Pointer< TimerTask > & task )`

Inserts the specified Task into the heap, heap is reordered to reflect the addition of a new element.

### Parameters

<i>task</i>	The TimerTask to insert into the heap.
-------------	--

**6.817.3.5** `bool decaf::internal::util::TimerTaskHeap::isEmpty ( ) const`

#### Returns

true if the heap is empty.

**6.817.3.6** `Pointer<TimerTask> decaf::internal::util::TimerTaskHeap::peek ( )`

Peaks at the Head of the Heap, returns the task with the nearest scheduled run time.

#### Returns

The TimerTask that is scheduled to be executed next if the Heap is empty a Null Pointer value is returned.

**6.817.3.7** `void decaf::internal::util::TimerTaskHeap::remove ( std::size_t pos )`

Removes the Task at the specified position from the heap, resorts the heap from the position down to the bottom.

#### Parameters

<i>pos</i>	The position at which to remove the TimerTask and begin a resort of the heap.
------------	---

**6.817.3.8** `void decaf::internal::util::TimerTaskHeap::reset ( )`

Clear all contents from the heap.

**6.817.3.9** `std::size_t decaf::internal::util::TimerTaskHeap::size ( ) const`

#### Returns

the size of the heap.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/TimerTaskHeap.h`

## 6.818 decaf::util::concurrent::TimeUnit Class Reference

A **TimeUnit** (p. 3919) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.

```
#include <src/main/decaf/util/concurrent/TimeUnit.h>
```

Inheritance diagram for decaf::util::concurrent::TimeUnit:

## Public Member Functions

- virtual **~TimeUnit** ()
- long long **convert** (long long sourceDuration, const **TimeUnit** &sourceUnit) const  
*Convert the given time duration in the given unit to this unit.*
- long long **toNanos** (long long duration) const  
*Equivalent to NANoseconds.convert(duration, this).*
- long long **toMicros** (long long duration) const  
*Equivalent to MICROseconds.convert(duration, this).*
- long long **toMillis** (long long duration) const  
*Equivalent to MILLIseconds.convert(duration, this).*
- long long **toSeconds** (long long duration) const  
*Equivalent to SECONDS.convert(duration, this).*
- long long **toMinutes** (long long duration) const  
*Equivalent to MINUTES.convert(duration, this).*
- long long **toHours** (long long duration) const  
*Equivalent to HOURS.convert(duration, this).*
- long long **toDays** (long long duration) const  
*Equivalent to DAYS.convert(duration, this).*
- void **timedWait** (**Synchronizable** \*obj, long long timeout) const throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException )  
*Perform a timed Object.wait using this time unit.*
- void **timedJoin** (decaf::lang::Thread \*thread, long long timeout) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException )  
*Perform a timed Thread.join using this time unit.*
- void **sleep** (long long timeout) const throw ( decaf::lang::exceptions::InterruptedException )  
*Perform a Thread.sleep using this unit.*

- virtual std::string **toString** () const  
*Converts the **TimeUnit** (p. 3919) type to the Name of the **TimeUnit** (p. 3919).*
- virtual int **compareTo** (const **TimeUnit** &value) const  
*Compares this object with the specified object for order.*
- virtual bool **equals** (const **TimeUnit** &value) const
- virtual bool **operator==** (const **TimeUnit** &value) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **TimeUnit** &value) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*

### Static Public Member Functions

- static const **TimeUnit** & **valueOf** (const std::string &name) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Returns the **TimeUnit** (p. 3919) constant of this type with the specified name.*

### Static Public Attributes

- static const **TimeUnit** **NANOSECONDS**  
*The Actual **TimeUnit** (p. 3919) enumerations.*
- static const **TimeUnit** **MICROSECONDS**
- static const **TimeUnit** **MILLISECONDS**
- static const **TimeUnit** **SECONDS**
- static const **TimeUnit** **MINUTES**
- static const **TimeUnit** **HOURS**
- static const **TimeUnit** **DAYS**
- static const **TimeUnit** \*const **values** []  
*The An Array of **TimeUnit** (p. 3919) Instances.*

### Protected Member Functions

- **TimeUnit** (int index, const std::string &name)  
*Hidden Constructor; this class can not be instantiated directly.*

### 6.818.1 Detailed Description

A **TimeUnit** (p. 3919) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units. A **TimeUnit** (p. 3919) does not maintain time information, but only helps organize and use time representations that may be maintained separately across various contexts. A nanosecond is defined as one thousandth of a microsecond, a microsecond as one thousandth of a millisecond, a millisecond as one thousandth of a second, a minute as sixty seconds, an hour as sixty minutes, and a day as twenty four hours.

A **TimeUnit** (p. 3919) is mainly used to inform time-based methods how a given timing parameter should be interpreted. For example, the following code will timeout in 50 milliseconds if the lock is not available:

```
Lock (p. 2450) lock = ...; if ( lock.tryLock( 50, TimeUnit.MILLISECONDS (p. 3929) ) ) ...
```

while this code will timeout in 50 seconds:

```
Lock (p. 2450) lock = ...; if ( lock.tryLock( 50, TimeUnit.SECONDS (p. 3930) ) ) ...
```

Note however, that there is no guarantee that a particular timeout implementation will be able to notice the passage of time at the same granularity as the given **TimeUnit** (p. 3919).

### 6.818.2 Constructor & Destructor Documentation

**6.818.2.1** `decaf::util::concurrent::TimeUnit::TimeUnit ( int index, const std::string & name )`  
[protected]

Hidden Constructor, this class can not be instantiated directly.

#### Parameters

<i>index</i>	- Index into the Time Unit set.
<i>name</i>	- Name of the unit type being represented.

**6.818.2.2** `virtual decaf::util::concurrent::TimeUnit::~~TimeUnit ( )` [inline, virtual]

### 6.818.3 Member Function Documentation

**6.818.3.1** `virtual int decaf::util::concurrent::TimeUnit::compareTo ( const TimeUnit & value )`  
`const` [virtual]

Compares this object with the specified object for order.

Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

In the foregoing description, the notation `sgn(expression)` designates the mathematical signum function, which is defined to return one of -1, 0, or 1 according to whether the value of expression is negative, zero or positive. The implementor must ensure `sgn(x.compareTo(y)) == -sgn(y.compareTo(x))` for all `x` and `y`. (This implies that `x.compareTo(y)` must throw an exception iff `y.compareTo(x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `(x.compareTo(y)>0 && y.compareTo(z)>0)` implies `x.compareTo(z)>0`.

Finally, the implementor must ensure that `x.compareTo(y)==0` implies that `sgn(x.compareTo(z)) == sgn(y.compareTo(z))`, for all `z`.

It is strongly recommended, but not strictly required that `(x.compareTo(y)==0) == (x.equals(y))`. Generally speaking, any class that implements the `Comparable` interface and violates this condition should clearly indicate this fact. The recommended language is "Note: this class has a natural ordering that is inconsistent with equals."

#### Parameters

<i>value</i>	- the Object to be compared.
--------------	------------------------------

#### Returns

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

#### 6.818.3.2 long long decaf::util::concurrent::TimeUnit::convert ( long long *sourceDuration*, const *TimeUnit* & *sourceUnit* ) const

Convert the given time duration in the given unit to this unit.

Conversions from finer to coarser granularities truncate, so lose precision. For example converting 999 milliseconds to seconds results in 0. Conversions from coarser to finer granularities with arguments that would numerically overflow saturate to `Long.MIN_VALUE` if negative or `Long.MAX_VALUE` if positive.

For example, to convert 10 minutes to milliseconds, use: `TimeUnit.MILLISECONDS.convert(10L, TimeUnit.MINUTES)` (p. 3929))

#### Parameters

<i>sourceDuration</i>	- Duration value to convert.
<i>sourceUnit</i>	- Unit type of the source duration.

#### Returns

the converted duration in this unit, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

**6.818.3.3** `virtual bool decaf::util::concurrent::TimeUnit::equals ( const TimeUnit & value )`  
`const` [virtual]

#### Returns

true if this value is considered equal to the passed value.

**6.818.3.4** `virtual bool decaf::util::concurrent::TimeUnit::operator< ( const TimeUnit & value )`  
`const` [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

#### Returns

true if this object is equal to the one passed.

**6.818.3.5** `virtual bool decaf::util::concurrent::TimeUnit::operator== ( const TimeUnit & value ) const` [virtual]

Compares equality between this object and the one passed.

#### Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

#### Returns

true if this object is equal to the one passed.

**6.818.3.6** `void decaf::util::concurrent::TimeUnit::sleep ( long long timeout ) const throw ( decaf::lang::exceptions::InterruptedException )`

Perform a `Thread.sleep` using this unit.

This is a convenience method that converts time arguments into the form required by the `Thread.sleep` method.

#### Parameters

<i>timeout</i>	the minimum time to sleep
----------------	---------------------------



**See also**

Thread::sleep

**6.818.3.7** void decaf::util::concurrent::TimeUnit::timedJoin ( decaf::lang::Thread \* *thread*, long long *timeout* ) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException )

Perform a timed Thread.join using this time unit.

This is a convenience method that converts time arguments into the form required by the Thread.join method.

**Parameters**

<i>thread</i>	the thread to wait for
<i>timeout</i>	the maximum time to wait

**Exceptions**

<i>InterruptedException</i>	if interrupted while waiting.
<i>NullPointerException</i>	if the thread object is null.

**See also**

Thread::join( long long, long long )

**6.818.3.8** void decaf::util::concurrent::TimeUnit::timedWait ( Synchronizable \* *obj*, long long *timeout* ) const throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException )

Perform a timed Object.wait using this time unit.

This is a convenience method that converts timeout arguments into the form required by the Object.wait method.

For example, you could implement a blocking poll method (see **BlockingQueue.poll** (p. 854)) using:

```
Object poll( long long timeout, const TimeUnit& unit )
    throw( InterruptedException ) {

    while( empty ) {
        unit.timedWait(this, timeout);
        ...
    }
}
```

**Parameters**

<i>obj</i>	the object to wait on
<i>timeout</i>	the maximum time to wait.

**Exceptions**

<i>InterruptedException</i>	if interrupted while waiting.
<i>NullPointerException</i>	if the <b>Synchronizable</b> (p. 3811) object is null.

**See also**

`Synchronizable::wait( long long, long long )`

**6.818.3.9** `long long decaf::util::concurrent::TimeUnit::toDays ( long long duration ) const`  
`[inline]`

Equivalent to `DAYS.convert( duration, this )`.

**Parameters**

<i>duration</i>	the duration
-----------------	--------------

**Returns**

the converted duration.

**See also**

**convert** (p. 3923)

**6.818.3.10** `long long decaf::util::concurrent::TimeUnit::toHours ( long long duration ) const`  
`[inline]`

Equivalent to `HOURS.convert( duration, this )`.

**Parameters**

<i>duration</i>	the duration
-----------------	--------------

**Returns**

the converted duration.

**See also**

**convert** (p. 3923)

**6.818.3.11** `long long decaf::util::concurrent::TimeUnit::toMicros ( long long duration ) const`  
[inline]

Equivalent to `MICROSECONDS.convert (duration, this)`.

#### Parameters

<i>duration</i>	the duration
-----------------	--------------

#### Returns

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

#### See also

`convert` (p. 3923)

**6.818.3.12** `long long decaf::util::concurrent::TimeUnit::toMillis ( long long duration ) const`  
[inline]

Equivalent to `MILLISECONDS.convert (duration, this)`.

#### Parameters

<i>duration</i>	the duration
-----------------	--------------

#### Returns

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

#### See also

`convert` (p. 3923)

**6.818.3.13** `long long decaf::util::concurrent::TimeUnit::toMinutes ( long long duration ) const`  
[inline]

Equivalent to `MINUTES.convert (duration, this)`.

#### Parameters

<i>duration</i>	the duration
-----------------	--------------

#### Returns

the converted duration.

**See also****convert** (p. 3923)

**6.818.3.14** `long long decaf::util::concurrent::TimeUnit::toNanos ( long long duration ) const`  
`[inline]`

Equivalent to `NANOSECONDS.convert (duration, this)`.

**Parameters**

<i>duration</i>	the duration
-----------------	--------------

**Returns**

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

**See also****convert** (p. 3923)

**6.818.3.15** `long long decaf::util::concurrent::TimeUnit::toSeconds ( long long duration ) const`  
`[inline]`

Equivalent to `SECONDS.convert (duration, this)`.

**Parameters**

<i>duration</i>	the duration
-----------------	--------------

**Returns**

the converted duration.

**See also****convert** (p. 3923)

**6.818.3.16** `virtual std::string decaf::util::concurrent::TimeUnit::toString ( ) const`  
`[virtual]`

Converts the **TimeUnit** (p. 3919) type to the Name of the **TimeUnit** (p. 3919).

**Returns**

String name of the **TimeUnit** (p. 3919)

```
6.818.3.17 static const TimeUnit& decaf::util::concurrent::TimeUnit::valueOf
( const std::string & name ) throw ( de-
caf::lang::exceptions::IllegalArgumentException )
[static]
```

Returns the **TimeUnit** (p. 3919) constant of this type with the specified name.

The string must match exactly an identifier used to declare an **TimeUnit** (p. 3919) constant in this type. (Extraneous whitespace characters are not permitted.)

#### Parameters

<i>name</i>	The Name of the <b>TimeUnit</b> (p. 3919) constant to be returned.
-------------	--

#### Returns

A constant reference to the **TimeUnit** (p. 3919) Constant with the given name.

#### Exceptions

<i>IllegalArgumentException</i>	if this enum type has no constant with the specified name
---------------------------------	---

### 6.818.4 Field Documentation

```
6.818.4.1 const TimeUnit decaf::util::concurrent::TimeUnit::DAYS [static]
```

```
6.818.4.2 const TimeUnit decaf::util::concurrent::TimeUnit::HOURS
[static]
```

```
6.818.4.3 const TimeUnit decaf::util::concurrent::TimeUnit::MICROSECONDS
[static]
```

```
6.818.4.4 const TimeUnit decaf::util::concurrent::TimeUnit::MILLISECONDS
[static]
```

```
6.818.4.5 const TimeUnit decaf::util::concurrent::TimeUnit::MINUTES
[static]
```

```
6.818.4.6 const TimeUnit decaf::util::concurrent::TimeUnit::NANOSECONDS
[static]
```

The Actual **TimeUnit** (p. 3919) enumerations.

**6.818.4.7** `const TimeUnit decaf::util::concurrent::TimeUnit::SECONDS`  
`[static]`

**6.818.4.8** `const TimeUnit* const decaf::util::concurrent::TimeUnit::values[]`  
`[static]`

The An Array of **TimeUnit** (p. 3919) Instances.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/TimeUnit.h`

## 6.819 cms::Topic Class Reference

An interface encapsulating a provider-specific topic name.

```
#include <src/main/cms/Topic.h>
```

Inheritance diagram for cms::Topic:

### Public Member Functions

- `virtual ~Topic ()`
- `virtual std::string getTopicName () const =0 throw ( CMSException )`

*Gets the name of this topic.*

### 6.819.1 Detailed Description

An interface encapsulating a provider-specific topic name. A **Topic** (p. 3930) is a Publish / Subscribe type **Destination** (p. 1776). All Messages sent to a **Topic** (p. 3930) are broadcast to all Subscribers of that **Topic** (p. 3930) unless the Subscriber defines a **Message** (p. 2614) selector that filters out that **Message** (p. 2614).

Since

1.0

## 6.819.2 Constructor & Destructor Documentation

6.819.2.1 virtual cms::Topic::~Topic ( ) [inline, virtual]

## 6.819.3 Member Function Documentation

6.819.3.1 virtual std::string cms::Topic::getTopicName ( ) const throw ( CMSEException )  
[pure virtual]

Gets the name of this topic.

### Returns

The topic name.

### Exceptions

<b>CMSEException</b> (p. 1190)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::commands::ActiveMQTopic** (p. 700).

The documentation for this class was generated from the following file:

- src/main/cms/**Topic.h**

## 6.820 activemq::state::Tracked Class Reference

```
#include <src/main/activemq/state/Tracked.h>
```

Inheritance diagram for activemq::state::Tracked:

### Public Member Functions

- **Tracked** ()
- **Tracked** (const **Pointer**< **decaf::lang::Runnable** > &runnable)
- virtual **~Tracked** ()
- void **onResponse** ()
- bool **isWaitingForResponse** () const

### 6.820.1 Constructor & Destructor Documentation

6.820.1.1 `activemq::state::Tracked::Tracked ( ) [inline]`

6.820.1.2 `activemq::state::Tracked::Tracked ( const Pointer< decaf::lang::Runnable > &runnable )`

6.820.1.3 `virtual activemq::state::Tracked::~~Tracked ( ) [inline, virtual]`

### 6.820.2 Member Function Documentation

6.820.2.1 `bool activemq::state::Tracked::isWaitingForResponse ( ) const [inline]`

6.820.2.2 `void activemq::state::Tracked::onResponse ( )`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/Tracked.h`

## 6.821 activemq::commands::TransactionId Class Reference

```
#include <src/main/activemq/commands/TransactionId.h>
```

Inheritance diagram for `activemq::commands::TransactionId`:

### Public Types

- `typedef decaf::lang::PointerComparator< TransactionId > COMPARATOR`

### Public Member Functions

- `TransactionId ()`
- `TransactionId (const TransactionId &other)`
- `virtual ~TransactionId ()`
- `virtual unsigned char getDataStructureType () const`  
*Get the unique identifier that this object and its own Marshaler share.*
- `virtual TransactionId * cloneDataStructure () const`  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- `virtual void copyDataStructure (const DataStructure *src)`  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*



- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual int **compareTo** (const **TransactionId** &value) const
- virtual bool **equals** (const **TransactionId** &value) const
- virtual bool **operator==** (const **TransactionId** &value) const
- virtual bool **operator<** (const **TransactionId** &value) const
- **TransactionId** & **operator=** (const **TransactionId** &other)

### Static Public Attributes

- static const unsigned char **ID\_TRANSACTIONID** = 0

### 6.821.1 Member Typedef Documentation

6.821.1.1 `typedef decaf::lang::PointerComparator<TransactionId>  
activemq::commands::TransactionId::COMPARATOR`

Reimplemented in `activemq::commands::LocalTransactionId` (p. 2421), and `activemq::commands::XATransactionId` (p. 4144).

### 6.821.2 Constructor & Destructor Documentation

6.821.2.1 `activemq::commands::TransactionId::TransactionId ( )`

6.821.2.2 `activemq::commands::TransactionId::TransactionId ( const TransactionId & other )`

6.821.2.3 `virtual activemq::commands::TransactionId::~~TransactionId ( ) [virtual]`

### 6.821.3 Member Function Documentation

6.821.3.1 `virtual TransactionId* activemq::commands::TransactionId::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

Reimplemented in **activemq::commands::LocalTransactionId** (p. 2422), and **activemq::commands::XATransactionId** (p. 4144).

**6.821.3.2** `virtual int activemq::commands::TransactionId::compareTo ( const TransactionId & value ) const` [virtual]

**6.821.3.3** `virtual void activemq::commands::TransactionId::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1715).

Reimplemented in **activemq::commands::LocalTransactionId** (p. 2422), and **activemq::commands::XATransactionId** (p. 4145).

**6.821.3.4** `virtual bool activemq::commands::TransactionId::equals ( const DataStructure * value ) const` [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1716).

Reimplemented in **activemq::commands::LocalTransactionId** (p. 2422), and **activemq::commands::XATransactionId** (p. 4145).

**6.821.3.5** `virtual bool activemq::commands::TransactionId::equals ( const TransactionId & value ) const` [virtual]

**6.821.3.6** `virtual unsigned char activemq::commands::TransactionId::getDataStructureType ( ) const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1713) type copy.

## 6.822 activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller

### Class Reference

---

3939

Implements **activemq::commands::DataStructure** (p. 1717).

Reimplemented in **activemq::commands::LocalTransactionId** (p. 2423), and **activemq::commands::XATransactionId** (p. 4145).

6.821.3.7 `virtual bool activemq::commands::TransactionId::operator< ( const TransactionId & value ) const` [virtual]

6.821.3.8 `TransactionId& activemq::commands::TransactionId::operator= ( const TransactionId & other )`

6.821.3.9 `virtual bool activemq::commands::TransactionId::operator== ( const TransactionId & value ) const` [virtual]

6.821.3.10 `virtual std::string activemq::commands::TransactionId::toString ( ) const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 841).

Reimplemented in **activemq::commands::LocalTransactionId** (p. 2424), and **activemq::commands::XATransactionId** (p. 4146).

### 6.821.4 Field Documentation

6.821.4.1 `const unsigned char activemq::commands::TransactionId::ID_TRANSACTIONID = 0` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/TransactionId.h`

## 6.822 activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller

### Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3935).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller**:

## Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshall an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.822.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3935). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.822.2 Constructor & Destructor Documentation

- 6.822.2.1 `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::TransactionIdMarshaller ( ) [inline]`
- 6.822.2.2 `virtual activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::~~TransactionIdMarshaller ( ) [inline, virtual]`

## 6.822.3 Member Function Documentation

- 6.822.3.1 `virtual void activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1675).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 2439), and `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 4170).

- 6.822.3.2 `virtual void activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller** (p. 2440), and **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller** (p. 4170).

```
6.822.3.3  virtual int activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller** (p. 2440), and **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller** (p. 4171).

```
6.822.3.4  virtual void activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

## 6.823 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller

### Class Reference

3943

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller** (p. 2441), and **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller** (p. 4171).

**6.822.3.5** virtual void **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::tightUnmarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream**  
\* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller** (p. 2441), and **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller** (p. 4172).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**TransactionIdMarshaller.h**

## 6.823 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller

### Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3939).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller`:

## Public Member Functions

- **TransactionIdMarshaller** ()
- virtual `~TransactionIdMarshaller` ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.823.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3939). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module



## 6.823.2 Constructor & Destructor Documentation

- 6.823.2.1 `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::TransactionIdMarshaller ( ) [inline]`
- 6.823.2.2 `virtual activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::~~TransactionIdMarshaller ( ) [inline, virtual]`

## 6.823.3 Member Function Documentation

- 6.823.3.1 `virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1675).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller` (p. 2448), and `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 4161).

- 6.823.3.2 `virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 2448), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 4162).

```
6.823.3.3  virtual int activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 2449), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 4162).

```
6.823.3.4  virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

## 6.824 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller

### Class Reference

3947

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 2449), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 4163).

```
6.823.3.5 virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 2450), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 4163).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h

## 6.824 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller

### Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3943).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller`:

## Public Member Functions

- **TransactionIdMarshaller** ()
- virtual `~TransactionIdMarshaller` ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.824.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3943). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.824.2 Constructor & Destructor Documentation

- 6.824.2.1 `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::TransactionIdMarshaller ( ) [inline]`
- 6.824.2.2 `virtual activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::~~TransactionIdMarshaller ( ) [inline, virtual]`

## 6.824.3 Member Function Documentation

- 6.824.3.1 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1675).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller` (p. 2431), and `activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 4153).

- 6.824.3.2 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 2431), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 4153).

**6.824.3.3** `virtual int activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 2432), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 4154).

**6.824.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

## 6.825 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller

### Class Reference

3951

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 2432), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 4154).

```
6.824.3.5 virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 2433), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 4155).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**TransactionIdMarshaller.h**

## 6.825 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller

### Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3947).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller`:

## Public Member Functions

- **TransactionIdMarshaller** ()
- virtual  $\sim$ **TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.825.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3947). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module



## 6.825.2 Constructor & Destructor Documentation

- 6.825.2.1 `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::TransactionIdMarshaller ( ) [inline]`
- 6.825.2.2 `virtual activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::~~TransactionIdMarshaller ( ) [inline, virtual]`

## 6.825.3 Member Function Documentation

- 6.825.3.1 `virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1675).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller` (p. 2435), and `activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 4166).

- 6.825.3.2 `virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 2435), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 4166).

**6.825.3.3** `virtual int activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 2436), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 4167).

**6.825.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

## 6.826 activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller

### Class Reference

3955

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 2436), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 4167).

```
6.825.3.5 virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 2437), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 4168).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**TransactionIdMarshaller.h**

## 6.826 activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller

### Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3951).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller`:

## Public Member Functions

- **TransactionIdMarshaller** ()
- virtual `~TransactionIdMarshaller` ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.826.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3951). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.826.2 Constructor & Destructor Documentation

- 6.826.2.1 `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::TransactionIdMarshaller ( ) [inline]`
- 6.826.2.2 `virtual activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::~~TransactionIdMarshaller ( ) [inline, virtual]`

## 6.826.3 Member Function Documentation

- 6.826.3.1 `virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1675).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller` (p. 2443), and `activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 4157).

- 6.826.3.2 `virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller** (p. 2444), and **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller** (p. 4158).

**6.826.3.3** `virtual int activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller** (p. 2444), and **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller** (p. 4158).

**6.826.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

## 6.827 activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller

### Class Reference

3959

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller** (p. 2445), and **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller** (p. 4159).

```
6.826.3.5 virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller** (p. 2445), and **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller** (p. 4159).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**TransactionIdMarshaller.h**

## 6.827 activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller

### Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3955).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/TransactionIdMarshaller.
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller`:

## Public Member Functions

- **TransactionIdMarshaller** ()
- virtual `~TransactionIdMarshaller` ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.827.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3955). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module



## 6.827.2 Constructor & Destructor Documentation

- 6.827.2.1 `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::TransactionIdMarshaller ( ) [inline]`
- 6.827.2.2 `virtual activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::~~TransactionIdMarshaller ( ) [inline, virtual]`

## 6.827.3 Member Function Documentation

- 6.827.3.1 `virtual void activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1675).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller` (p. 2426), and `activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller` (p. 4149).

- 6.827.3.2 `virtual void activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller** (p. 2427), and **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller** (p. 4149).

```
6.827.3.3  virtual int activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller** (p. 2427), and **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller** (p. 4150).

```
6.827.3.4  virtual void activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller** (p. 2428), and **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller** (p. 4150).

```
6.827.3.5 virtual void activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller** (p. 2428), and **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller** (p. 4151).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/TransactionIdMarshaller.h

**6.828 activemq::commands::TransactionInfo Class Reference**

```
#include <src/main/activemq/commands/TransactionInfo.h>
```

Inheritance diagram for **activemq::commands::TransactionInfo**:

## Public Member Functions

- **TransactionInfo** ()
- virtual **~TransactionInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **TransactionInfo** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
- virtual unsigned char **getType** () const
- virtual void **setType** (unsigned char type)
- virtual bool **isTransactionInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

## Static Public Attributes

- static const unsigned char **ID\_TRANSACTIONINFO** = 7

## Protected Attributes

- **Pointer**< **ConnectionId** > **connectionId**
- **Pointer**< **TransactionId** > **transactionId**
- unsigned char **type**

## 6.828.1 Constructor & Destructor Documentation

6.828.1.1 `activemq::commands::TransactionInfo::TransactionInfo ( )`

6.828.1.2 `virtual activemq::commands::TransactionInfo::~~TransactionInfo ( ) [virtual]`

## 6.828.2 Member Function Documentation

6.828.2.1 `virtual TransactionInfo* activemq::commands::TransactionInfo::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1714).

6.828.2.2 `virtual void activemq::commands::TransactionInfo::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

6.828.2.3 `virtual bool activemq::commands::TransactionInfo::equals ( const DataStructure * value ) const [virtual]`

Compares the `DataStructure` (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 766).

6.828.2.4 **virtual const Pointer<ConnectionId>& activemq::commands::TransactionInfo::getConnectionId ( ) const**  
[virtual]

6.828.2.5 **virtual Pointer<ConnectionId>& activemq::commands::TransactionInfo::getConnectionId ( )**  
[virtual]

6.828.2.6 **virtual unsigned char activemq::commands::TransactionInfo::getDataStructureType ( ) const** [virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataSet** (p. 1713) type copy.

Implements **activemq::commands::DataSet** (p. 1717).

6.828.2.7 **virtual const Pointer<TransactionId>& activemq::commands::TransactionInfo::getTransactionId ( ) const**  
[virtual]

6.828.2.8 **virtual Pointer<TransactionId>& activemq::commands::TransactionInfo::getTransactionId ( )**  
[virtual]

6.828.2.9 **virtual unsigned char activemq::commands::TransactionInfo::getType ( ) const**  
[virtual]

6.828.2.10 **virtual bool activemq::commands::TransactionInfo::isTransactionInfo ( ) const**  
[inline, virtual]

#### Returns

an answer of true to the **isTransactionInfo()** (p. 3962) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

- 6.828.2.11 `virtual void activemq::commands::TransactionInfo::setConnectionId ( const Pointer< ConnectionId > & connectionId ) [virtual]`
- 6.828.2.12 `virtual void activemq::commands::TransactionInfo::setTransactionId ( const Pointer< TransactionId > & transactionId ) [virtual]`
- 6.828.2.13 `virtual void activemq::commands::TransactionInfo::setType ( unsigned char type ) [virtual]`
- 6.828.2.14 `virtual std::string activemq::commands::TransactionInfo::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

- 6.828.2.15 `virtual Pointer<Command> activemq::commands::TransactionInfo::visit ( activemq::state::CommandVisitor * visitor ) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

### 6.828.3 Field Documentation

- 6.828.3.1 **Pointer<ConnectionId> activemq::commands::TransactionInfo::connectionId**  
[protected]
- 6.828.3.2 **const unsigned char activemq::commands::TransactionInfo::ID\_-TRANSACTIONINFO = 7** [static]
- 6.828.3.3 **Pointer<TransactionId> activemq::commands::TransactionInfo::transactionId**  
[protected]
- 6.828.3.4 **unsigned char activemq::commands::TransactionInfo::type**  
[protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**TransactionInfo.h**

## 6.829 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3964).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/TransactionInfo
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller:

### Public Member Functions

- **TransactionInfoMarshaller ()**
- virtual **~TransactionInfoMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)**  
throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )



## 6.829

**activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller**

**Class Reference**

**3969**

---

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.829.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3964).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.829.2 Constructor & Destructor Documentation

**6.829.2.1** **activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::TransactionInfoMarshaller**  
( ) [inline]

**6.829.2.2** **virtual activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::~~TransactionInfoMarshaller**  
( ) [inline, virtual]

### 6.829.3 Member Function Documentation

**6.829.3.1** **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::createObject** (  
) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.829.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.829.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 794).

**6.829.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

6.829

**activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller**

**Class Reference**

**3971**

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 795).

**6.829.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 796).

**6.829.3.6** `virtual void activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 797).

```

6.829.3.7 virtual void activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 799).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**TransactionInfoMarshaller.h**

## 6.830 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3968).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfo
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller**:

#### Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

## 6.830

**activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller**

### Class Reference

3973

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.830.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p.3968).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.830.2 Constructor & Destructor Documentation

6.830.2.1 **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::TransactionInfoMarshaller**  
( ) [inline]

6.830.2.2 **virtual activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::~~TransactionInfoMarshaller**  
( ) [inline, virtual]

### 6.830.3 Member Function Documentation

6.830.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.830.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.830.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 787).

**6.830.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

6.830

activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller

Class Reference

3975

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 788).

```
6.830.3.5  virtual int activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 789).

```
6.830.3.6  virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 790).

```
6.830.3.7 virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 792).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**TransactionInfoMarshaller.h**

## 6.831 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3972).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller**:



- **TransactionInfoMarshaller ()**
- virtual **~TransactionInfoMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.831.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3972).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.831.2 Constructor & Destructor Documentation

6.831.2.1 `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::TransactionInfoMarshaller ( ) [inline]`

6.831.2.2 `virtual activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::~~TransactionInfoMarshaller ( ) [inline, virtual]`

## 6.831.3 Member Function Documentation

6.831.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.831.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.831.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

6.831

**activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller****Class Reference****3979****Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 772).

**6.831.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 774).

**6.831.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 775).

```
6.831.3.6 virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 776).

```
6.831.3.7 virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 777).

The documentation for this class was generated from the following file:

6.832

**activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller**

**Class Reference**

3981

- `src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h`

## 6.832 **activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller** **Class Reference**

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3977).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/TransactionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller`:

### Public Member Functions

- **TransactionInfoMarshaller ()**
- virtual **~TransactionInfoMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn)** throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut)** throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.832.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3977).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.832.2 Constructor & Destructor Documentation

6.832.2.1 **activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::TransactionInfoMarshaller**  
 ( ) [inline]

6.832.2.2 **virtual activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::~~TransactionInfoMarshaller**  
 ( ) [inline, virtual]

### 6.832.3 Member Function Documentation

6.832.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::createObject (**  
**) const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.832.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::getDataStructureType**  
**( ) const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.832.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure**  
**\* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw (**  
**decaf::io::IOException )** [virtual]

Write a object instance to data output stream.

6.832

**activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller**

**Class Reference**

3983

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 801).

**6.832.3.4** virtual void **activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 802).

**6.832.3.5** virtual int **activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::tightMarshal1** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 803).

```
6.832.3.6 virtual void activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 804).

```
6.832.3.7 virtual void activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.



6.833

**activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller**

**Class Reference**

**3985**

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 806).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**TransactionInfoMarshaller.h**

## 6.833 **activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3981).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller**:

### Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.833.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3981).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.833.2 Constructor & Destructor Documentation

6.833.2.1 **activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::TransactionInfoMarshaller**  
( ) [inline]

6.833.2.2 **virtual activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::~~TransactionInfoMarshaller**  
( ) [inline, virtual]

### 6.833.3 Member Function Documentation

6.833.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::createObject** (  
) **const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.833.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

6.833

**activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller**

**Class Reference**

**3987**

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.833.3.3** virtual void **activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::looseMarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 780).

**6.833.3.4** virtual void **activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 781).

```

6.833.3.5  virtual int activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 782).

```

6.833.3.6  virtual void activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 783).

6.834

activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller

Class Reference

3989

6.833.3.7 virtual void activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::tightUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream  
\* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**TransactionInfoMarshaller.h**

## 6.834 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3985).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/TransactionInfoMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller**:

#### Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.834.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p.3985).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.834.2 Constructor & Destructor Documentation

6.834.2.1 **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::TransactionInfoMarshaller**  
( ) [inline]

6.834.2.2 **virtual activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::~~TransactionInfoMarshaller**  
( ) [inline, virtual]

### 6.834.3 Member Function Documentation

6.834.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

6.834

**activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller**

**Class Reference**

**3991**

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.834.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.834.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 808).

**6.834.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 809).

```
6.834.3.5  virtual int activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 810).

```
6.834.3.6  virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled



<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 811).

**6.834.3.7** virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 813).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**TransactionInfoMarshaller.h**

**6.835 activemq::state::TransactionState Class Reference**

```
#include <src/main/activemq/state/TransactionState.h>
```

**Public Member Functions**

- **TransactionState** (const **Pointer**< **TransactionId** > &id)
- virtual ~**TransactionState** ()
- std::string **toString** () const
- void **addCommand** (const **Pointer**< **Command** > &operation)

- void **checkShutdown** () const
- void **shutdown** ()
- const **StlList**< **Pointer**< **Command** > > & **getCommands** () const
- const **Pointer**< **TransactionId** > & **getId** () const
- void **setPrepared** (bool prepared)
- bool **isPrepared** () const
- void **setPreparedResult** (int preparedResult)
- int **getPreparedResult** () const
- void **addProducerState** (const **Pointer**< **ProducerState** > &producerState)
- std::vector< **Pointer**< **ProducerState** > > **getProducerStates** ()

### 6.835.1 Constructor & Destructor Documentation

6.835.1.1 `activemq::state::TransactionState::TransactionState ( const Pointer< TransactionId > & id )`

6.835.1.2 `virtual activemq::state::TransactionState::~~TransactionState ( ) [virtual]`

### 6.835.2 Member Function Documentation

6.835.2.1 `void activemq::state::TransactionState::addCommand ( const Pointer< Command > & operation )`

6.835.2.2 `void activemq::state::TransactionState::addProducerState ( const Pointer< ProducerState > & producerState )`

6.835.2.3 `void activemq::state::TransactionState::checkShutdown ( ) const`

6.835.2.4 `const StlList< Pointer<Command> >& activemq::state::TransactionState::getCommands ( ) const [inline]`

6.835.2.5 `const Pointer<TransactionId>& activemq::state::TransactionState::getId ( ) const [inline]`

6.835.2.6 `int activemq::state::TransactionState::getPreparedResult ( ) const [inline]`

6.835.2.7 `std::vector< Pointer<ProducerState> > activemq::state::TransactionState::getProducerStates ( )`

6.835.2.8 `bool activemq::state::TransactionState::isPrepared ( ) const [inline]`

6.835.2.9 `void activemq::state::TransactionState::setPrepared ( bool prepared ) [inline]`

6.835.2.10 `void activemq::state::TransactionState::setPreparedResult ( int preparedResult ) [inline]`

6.835.2.11 `void activemq::state::TransactionState::shutdown ( ) [inline]`

6.835.2.12 `std::string activemq::state::TransactionState::toString ( ) const`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/TransactionState.h`

## 6.836 decaf::internal::util::concurrent::Transferer< E > Class Template Reference

Shared internal API for dual stacks and queues.

```
#include <src/main/decaf/internal/util/concurrent/Transferer.h>
```

Inheritance diagram for `decaf::internal::util::concurrent::Transferer< E >`:

### 6.836.1 Detailed Description

```
template<typename E> class decaf::internal::util::concurrent::Transferer< E >
```

Shared internal API for dual stacks and queues.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/concurrent/Transferer.h`

## 6.837 `decaf::internal::util::concurrent::TransferQueue< E >` Class Template Reference

This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.

```
#include <src/main/decaf/internal/util/concurrent/TransferQueue.h>
```

Inheritance diagram for `decaf::internal::util::concurrent::TransferQueue< E >`:

### Public Member Functions

- **TransferQueue** ()

*Node class for **TransferQueue** (p. 3992).*

- virtual **~TransferQueue** ()
- virtual void **transfer** (E \*e, bool timed, long long nanos) throw ( decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException )

*Performs a put.*

- virtual E \* **transfer** (bool timed, long long nanos) throw ( decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException )

*Performs a take.*

### 6.837.1 Detailed Description

template<typename E> class decaf::internal::util::concurrent::TransferQueue< E >

This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers. The algorithm is a little simpler than that for stacks because fulfillers do not need explicit nodes, and matching is done by CAS'ing QNode.item field from non-null to null (for put) or vice versa (for take).

### 6.837.2 Constructor & Destructor Documentation

6.837.2.1 template<typename E> decaf::internal::util::concurrent::TransferQueue< E>::TransferQueue ( ) [inline]

Node class for **TransferQueue** (p. 3992).

Tries to cancel by CAS'ing ref to NULL if that succeeds then we mark as cancelled. Returns true if this node is known to be off the queue because its next pointer has been forgotten due to an advanceHead operation.

6.837.2.2 template<typename E> virtual decaf::internal::util::concurrent::TransferQueue< E>::~~TransferQueue ( ) [inline, virtual]

### 6.837.3 Member Function Documentation

6.837.3.1 template<typename E> virtual void decaf::internal::util::concurrent::TransferQueue< E>::transfer ( E \* e, bool *timed*, long long *nanos* ) throw ( decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]

Performs a put.

#### Parameters

<i>e</i>	the item to be handed to a consumer;
<i>timed</i>	if this operation should timeout
<i>nanos</i>	the timeout, in nanoseconds

#### Exceptions

<i>TimeoutException</i>	if the operation timed out waiting for the consumer to accept the item offered.
<i>InterruptedException</i>	if the thread was interrupted while waiting for the consumer to accept the item offered.

Implements **decaf::internal::util::concurrent::Transferer< E >** (p. 3991).

```

6.837.3.2  template<typename E > virtual E*
           decaf::internal::util::concurrent::TransferQueue<
           E >::transfer ( bool timed, long long nanos ) throw
           ( decaf::util::concurrent::TimeoutException,
           decaf::lang::exceptions::InterruptedException ) [inline,
           virtual]

```

Performs a take.

#### Parameters

<i>timed</i>	if this operation should timeout
<i>nanos</i>	the timeout, in nanoseconds

#### Returns

the item provided or received;

#### Exceptions

<i>TimeoutException</i>	if the operation timed out waiting for the producer to offer an item.
<i>InterruptedException</i>	if the thread was interrupted while waiting for the producer to offer an item.

Implements **decaf::internal::util::concurrent::Transferer**< E > (p. 3991).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/**TransferQueue.h**

## 6.838 decaf::internal::util::concurrent::TransferStack< E > Class Template Reference

```
#include <src/main/decaf/internal/util/concurrent/TransferStack.h>
```

Inheritance diagram for decaf::internal::util::concurrent::TransferStack< E >:

#### Public Member Functions

- **TransferStack** ()
- virtual **~TransferStack** ()
- virtual void **transfer** (E \*e, bool timed, long long nanos) throw ( decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException )  
*Performs a put.*
- virtual E \* **transfer** (bool timed, long long nanos) throw ( decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException )

*Performs a take.*

```
template<typename E> class decaf::internal::util::concurrent::TransferStack< E >
```

## 6.838.1 Constructor & Destructor Documentation

6.838.1.1 `template<typename E> decaf::internal::util::concurrent::TransferStack< E >::TransferStack ( ) [inline]`

6.838.1.2 `template<typename E> virtual decaf::internal::util::concurrent::TransferStack< E >::~~TransferStack ( ) [inline, virtual]`

## 6.838.2 Member Function Documentation

6.838.2.1 `template<typename E> virtual void decaf::internal::util::concurrent::TransferStack< E >::transfer ( E * e, bool timed, long long nanos ) throw ( decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Performs a put.

### Parameters

<i>e</i>	the item to be handed to a consumer;
<i>timed</i>	if this operation should timeout
<i>nanos</i>	the timeout, in nanoseconds

### Exceptions

<i>TimeoutException</i>	if the operation timed out waiting for the consumer to accept the item offered.
<i>InterruptedException</i>	if the thread was interrupted while waiting for the consumer to accept the item offered.

Implements `decaf::internal::util::concurrent::Transferer< E >` (p. 3991).

6.838.2.2 `template<typename E> virtual E* decaf::internal::util::concurrent::TransferStack< E >::transfer ( bool timed, long long nanos ) throw ( decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Performs a take.

**Parameters**

<i>timed</i>	if this operation should timeout
<i>nanos</i>	the timeout, in nanoseconds

**Returns**

the item provided or received;

**Exceptions**

<i>TimeoutException</i>	if the operation timed out waiting for the producer to offer an item.
<i>InterruptedException</i>	if the thread was interrupted while waiting for the producer to offer an item.

Implements **decaf::internal::util::concurrent::Transferer< E >** (p. 3991).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/**TransferStack.h**

**6.839 activemq::transport::Transport Class Reference**

Interface for a transport layer for command objects.

```
#include <src/main/activemq/transport/Transport.h>
```

Inheritance diagram for activemq::transport::Transport:

**Public Member Functions**

- virtual **~Transport** ()
- virtual void **start** ()=0 throw ( decaf::io::IOException )  
*Starts the **Transport** (p. 3996), the send methods of a **Transport** (p. 3996) will throw an exception if used before the **Transport** (p. 3996) is started.*
- virtual void **stop** ()=0 throw ( decaf::io::IOException )  
*Stops the **Transport** (p. 3996).*
- virtual void **oneway** (const **Pointer**< **Command** > &command)=0 throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends a one-way command.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command)=0 throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )



*Sends the given command to the broker and then waits for the response.*

- virtual **Pointer< Response > request** (const **Pointer< Command > &command**, unsigned int timeout)=0 throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Sends the given command to the broker and then waits for the response.*

- virtual void **setWireFormat** (const **Pointer< wireformat::WireFormat > &wireFormat**)=0

*Sets the WireFormat instance to use.*

- virtual void **setTransportListener** (**TransportListener \*listener**)=0

*Sets the observer of asynchronous events from this transport.*

- virtual **TransportListener \* getTransportListener** () const =0

*Gets the observer of asynchronous events from this transport.*

- virtual **Transport \* narrow** (const std::type\_info &typeId)=0

*Narrows down a Chain of Transports to a specific **Transport** (p. 3996) to allow a higher level transport to skip intermediate Transports in certain circumstances.*

- virtual bool **isFaultTolerant** () const =0

*Is this **Transport** (p. 3996) fault tolerant, meaning that it will reconnect to a broker on disconnect.*

- virtual bool **isConnected** () const =0

*Is the **Transport** (p. 3996) Connected to its Broker.*

- virtual bool **isClosed** () const =0

*Has the **Transport** (p. 3996) been shutdown and no longer usable.*

- virtual std::string **getRemoteAddress** () const =0

- virtual void **reconnect** (const **decaf::net::URI &uri**)=0 throw ( decaf::io::IOException )

*reconnect to another location*

### 6.839.1 Detailed Description

Interface for a transport layer for command objects. Callers can send oneway messages or make synchronous requests. Non-response messages will be delivered to the specified listener object upon receipt. A user of the **Transport** (p. 3996) can set an exception listener to be notified of errors that occurs in Threads that the **Transport** (p. 3996) layer runs. Transports should be given an instance of a WireFormat object when created so that they can turn the built in Commands to / from the required wire format encoding.

## 6.839.2 Constructor & Destructor Documentation

6.839.2.1 `virtual activemq::transport::Transport::~~Transport ( ) [inline, virtual]`

## 6.839.3 Member Function Documentation

6.839.3.1 `virtual std::string activemq::transport::Transport::getRemoteAddress ( ) const [pure virtual]`

### Returns

the remote address for this connection

Implemented in `activemq::transport::failover::FailoverTransport` (p. 1934), `activemq::transport::IOTransport` (p. 2216), `activemq::transport::mock::MockTransport` (p. 2858), and `activemq::transport::TransportFilter` (p. 4008).

6.839.3.2 `virtual TransportListener* activemq::transport::Transport::getTransportListener ( ) const [pure virtual]`

Gets the observer of asynchronous events from this transport.

### Returns

the listener of transport events.

Implemented in `activemq::transport::failover::FailoverTransport` (p. 1935), `activemq::transport::IOTransport` (p. 2216), `activemq::transport::mock::MockTransport` (p. 2858), and `activemq::transport::TransportFilter` (p. 4008).

6.839.3.3 `virtual bool activemq::transport::Transport::isClosed ( ) const [pure virtual]`

Has the **Transport** (p. 3996) been shutdown and no longer usable.

### Returns

true if the **Transport** (p. 3996)

Implemented in `activemq::transport::failover::FailoverTransport` (p. 1935), `activemq::transport::IOTransport` (p. 2216), `activemq::transport::mock::MockTransport` (p. 2859), `activemq::transport::tcp::TcpTransport` (p. 3868), and `activemq::transport::TransportFilter` (p. 4008).

6.839.3.4 `virtual bool activemq::transport::Transport::isConnected ( ) const [pure virtual]`

Is the **Transport** (p. 3996) Connected to its Broker.

**Returns**

true if a connection has been made.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1936), **activemq::transport::IOTransport** (p. 2216), **activemq::transport::mock::MockTransport** (p. 2859), **activemq::transport::tcp::TcpTransport** (p. 3868), and **activemq::transport::TransportFilter** (p. 4008).

**6.839.3.5** `virtual bool activemq::transport::Transport::isFaultTolerant ( ) const` [pure virtual]

Is this **Transport** (p. 3996) fault tolerant, meaning that it will reconnect to a broker on disconnect.

**Returns**

true if the **Transport** (p. 3996) is fault tolerant.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1936), **activemq::transport::IOTransport** (p. 2217), **activemq::transport::mock::MockTransport** (p. 2859), **activemq::transport::tcp::TcpTransport** (p. 3868), and **activemq::transport::TransportFilter** (p. 4009).

Referenced by **activemq::transport::TransportFilter::isFaultTolerant()**.

**6.839.3.6** `virtual Transport* activemq::transport::Transport::narrow ( const std::type_info & typeId )` [pure virtual]

Narrows down a Chain of Transports to a specific **Transport** (p. 3996) to allow a higher level transport to skip intermediate Transports in certain circumstances.

**Parameters**

<i>typeId</i> - The type_info of the Object we are searching for.
---

**Returns**

the requested Object. or NULL if its not in this chain.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1937), **activemq::transport::IOTransport** (p. 2217), **activemq::transport::mock::MockTransport** (p. 2860), and **activemq::transport::TransportFilter** (p. 4009).

Referenced by **activemq::transport::failover::FailoverTransport::narrow()**.

**6.839.3.7** `virtual void activemq::transport::Transport::oneway ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )` [pure virtual]

Sends a one-way command.

Does not wait for any response from the broker.

#### Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

#### Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3386), **activemq::transport::failover::FailoverTransport** (p. 1937), **activemq::transport::inactivity::InactivityMonitor** (p. 2068), **activemq::transport::IOTransport** (p. 2217), **activemq::transport::logging::LoggingTransport** (p. 2479), **activemq::transport::mock::MockTransport** (p. 2860), **activemq::transport::TransportFilter** (p. 4010), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2987).

**6.839.3.8** `virtual void activemq::transport::Transport::reconnect ( const decaf::net::URI & uri ) throw ( decaf::io::IOException ) [pure virtual]`

reconnect to another location

#### Parameters

<i>uri</i>	
------------	--

#### Exceptions

<i>IOException</i>	on failure of if not supported
--------------------	--------------------------------

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1938), and **activemq::transport::TransportFilter** (p. 4010).

**6.839.3.9** `virtual Pointer<Response> activemq::transport::Transport::request ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [pure virtual]`

Sends the given command to the broker and then waits for the response.

#### Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

#### Returns

the response from the broker.

**Exceptions**

<i>IOException</i>	if an exception occurs during the read of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3387), **activemq::transport::failover::FailoverTransport** (p. 1939), **activemq::transport::IOTransport** (p. 2218), **activemq::transport::logging::LoggingTransport** (p. 2479), **activemq::transport::mock::MockTransport** (p. 2861), **activemq::transport::TransportFilter** (p. 4011), and **activemq::wireformat::openwire::OpenWireFormatNeg** (p. 2988).

**6.839.3.10** `virtual Pointer<Response> activemq::transport::Transport::request ( const Pointer< Command > & command, unsigned int timeout ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [pure virtual]`

Sends the given command to the broker and then waits for the response.

**Parameters**

<i>command</i>	- The command to be sent.
<i>timeout</i>	- The time to wait for this response.

**Returns**

the response from the broker.

**Exceptions**

<i>IOException</i>	if an exception occurs during the read of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3387), **activemq::transport::failover::FailoverTransport** (p. 1939), **activemq::transport::IOTransport** (p. 2218), **activemq::transport::logging::LoggingTransport** (p. 2479), **activemq::transport::mock::MockTransport** (p. 2861), **activemq::transport::TransportFilter** (p. 4011), and **activemq::wireformat::openwire::OpenWireFormatNeg** (p. 2989).

**6.839.3.11** `virtual void activemq::transport::Transport::setTransportListener ( TransportListener * listener ) [pure virtual]`

Sets the observer of asynchronous events from this transport.

**Parameters**

<i>listener</i>	the listener of transport events.
-----------------	-----------------------------------

Implemented in `activemq::transport::failover::FailoverTransport` (p. 1941), `activemq::transport::IOTransport` (p. 2219), `activemq::transport::mock::MockTransport` (p. 2863), and `activemq::transport::TransportFilter` (p. 4012).

**6.839.3.12** `virtual void activemq::transport::Transport::setWireFormat ( const Pointer< wireformat::WireFormat > & wireFormat ) [pure virtual]`

Sets the `WireFormat` instance to use.

#### Parameters

<i>wireFormat</i>	The <code>WireFormat</code> the object used to encode / decode commands.
-------------------	--

Implemented in `activemq::transport::IOTransport` (p. 2220), and `activemq::transport::TransportFilter` (p. 4012).

**6.839.3.13** `virtual void activemq::transport::Transport::start ( ) throw ( decaf::io::IOException ) [pure virtual]`

Starts the **Transport** (p. 3996), the send methods of a **Transport** (p. 3996) will throw an exception if used before the **Transport** (p. 3996) is started.

#### Exceptions

<i>IOException</i>	if an error occurs while starting the <b>Transport</b> (p. 3996).
--------------------	---

Implemented in `activemq::transport::correlator::ResponseCorrelator` (p. 3388), `activemq::transport::failover::FailoverTransport` (p. 1942), `activemq::transport::IOTransport` (p. 2220), `activemq::transport::mock::MockTransport` (p. 2863), `activemq::transport::TransportFilter` (p. 4012), and `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p. 2989).

**6.839.3.14** `virtual void activemq::transport::Transport::stop ( ) throw ( decaf::io::IOException ) [pure virtual]`

Stops the **Transport** (p. 3996).

#### Exceptions

<i>IOException</i>	if an error occurs while stopping the transport.
--------------------	--

Implemented in `activemq::transport::failover::FailoverTransport` (p. 1942), `activemq::transport::IOTransport` (p. 2220), `activemq::transport::mock::MockTransport` (p. 2864), and `activemq::transport::TransportFilter` (p. 4012).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/Transport.h`

## 6.840 activemq::transport::TransportFactory Class Reference

Defines the interface for Factories that create Transports or TransportFilters.

```
#include <src/main/activemq/transport/TransportFactory.h>
```

Inheritance diagram for activemq::transport::TransportFactory:

### Public Member Functions

- virtual **~TransportFactory** ()
- virtual **Pointer< Transport > create** (const **decaf::net::URI** &location)=0 throw ( exceptions::ActiveMQException )
 

*Creates a fully configured **Transport** (p. 3996) instance which could be a chain of filters and transports.*
- virtual **Pointer< Transport > createComposite** (const **decaf::net::URI** &location)=0 throw ( exceptions::ActiveMQException )
 

*Creates a slimed down **Transport** (p. 3996) instance which can be used in composite transport instances.*

### 6.840.1 Detailed Description

Defines the interface for Factories that create Transports or TransportFilters. The factory should be able to create either a completely configured **Transport** (p. 3996) meaning that it has all the appropriate filters wrapping it, or it should be able to create a slimed down version that is used in composite transports like Failover or Fanout.

Since

3.0

### 6.840.2 Constructor & Destructor Documentation

6.840.2.1 **virtual activemq::transport::TransportFactory::~TransportFactory** ( ) [inline, virtual]

### 6.840.3 Member Function Documentation

6.840.3.1 **virtual Pointer<Transport> activemq::transport::TransportFactory::create** ( const **decaf::net::URI** & location ) throw ( exceptions::ActiveMQException ) [pure virtual]

Creates a fully configured **Transport** (p. 3996) instance which could be a chain of filters and transports.

**Parameters**

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

**Exceptions**

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implemented in **activemq::transport::failover::FailoverTransportFactory** (p. 1944), **activemq::transport::mock::MockTransportFactory** (p. 2865), and **activemq::transport::tcp::TcpTransportFactory** (p. 3870).

**6.840.3.2** `virtual Pointer<Transport> activemq::transport::TransportFactory::createComposite ( const decaf::net::URI & location ) throw ( exceptions::ActiveMQException )`  
[pure virtual]

Creates a slimmed down **Transport** (p. 3996) instance which can be used in composite transport instances.

**Parameters**

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

**Exceptions**

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implemented in **activemq::transport::failover::FailoverTransportFactory** (p. 1944), **activemq::transport::mock::MockTransportFactory** (p. 2865), and **activemq::transport::tcp::TcpTransportFactory** (p. 3870).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/TransportFactory.h`

**6.841 activemq::transport::TransportFilter Class Reference**

A filter on the transport layer.

```
#include <src/main/activemq/transport/TransportFilter.h>
```

Inheritance diagram for `activemq::transport::TransportFilter`:

**Public Member Functions**

- **TransportFilter** (const **Pointer**< **Transport** > &next)



*Constructor.*

- virtual **~TransportFilter** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)  
*Event handler for the receipt of a command.*
- virtual void **onException** (const **decaf::lang::Exception** &ex)  
*Event handler for an exception from a command transport.*
- virtual void **transportInterrupted** ()  
*The transport has suffered an interruption from which it hopes to recover.*
- virtual void **transportResumed** ()  
*The transport has resumed after an interruption.*
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw ( **decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException** )

*Sends a one-way command.*

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw ( **decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException** )  
*Not supported by this class - throws an exception.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw ( **decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException** )  
*Not supported by this class - throws an exception.*
- virtual void **setTransportListener** (**TransportListener** \*listener)  
*Sets the observer of asynchronous exceptions from this transport.*
- virtual **TransportListener** \* **getTransportListener** () const  
*Gets the observer of asynchronous exceptions from this transport.*
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)  
*Sets the WireFormat instance to use.*
- virtual void **start** () throw ( **decaf::io::IOException** )  
*Starts this transport object and creates the thread for polling on the input stream for commands.*
- virtual void **stop** () throw ( **decaf::io::IOException** )  
*Stops the **Transport** (p. 3996).*

- virtual void **close** () throw ( decaf::io::IOException )  
*Stops the polling thread and closes the streams.*
- virtual **Transport \* narrow** (const std::type\_info &typeId)  
*Narrows down a Chain of Transports to a specific **Transport** (p. 3996) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () const  
*Is this **Transport** (p. 3996) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const  
*Is the **Transport** (p. 3996) Connected to its Broker.*
- virtual bool **isClosed** () const  
*Has the **Transport** (p. 3996) been shutdown and no longer usable.*
- virtual std::string **getRemoteAddress** () const
- virtual void **reconnect** (const decaf::net::URI &uri) throw ( decaf::io::IOException )  
*reconnect to another location*

### Protected Member Functions

- void **fire** (const decaf::lang::Exception &ex)  
*Notify the listener of the thrown Exception.*
- void **fire** (const **Pointer**< **Command** > &command)  
*Notify the listener of the new incoming Command.*

### Protected Attributes

- **Pointer**< **Transport** > **next**  
*The transport that this filter wraps around.*
- **TransportListener \* listener**  
*Listener of this transport.*

#### 6.841.1 Detailed Description

A filter on the transport layer. **Transport** (p. 3996) filters implement the **Transport** (p. 3996) interface and optionally delegate calls to another **Transport** (p. 3996) object.

Since

1.0

## 6.841.2 Constructor & Destructor Documentation

**6.841.2.1** `activemq::transport::TransportFilter::TransportFilter ( const Pointer< Transport > & next )`

Constructor.

### Parameters

<i>next</i>	- the next <b>Transport</b> (p. 3996) in the chain
-------------	--

**6.841.2.2** `virtual activemq::transport::TransportFilter::~~TransportFilter ( ) [inline, virtual]`

## 6.841.3 Member Function Documentation

**6.841.3.1** `virtual void activemq::transport::TransportFilter::close ( ) throw ( decaf::io::IOException ) [virtual]`

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

### Exceptions

<i>IOException</i>	if an error occurs while closing the <b>Transport</b> (p. 3996).
--------------------	--

Implements **decaf::io::Closeable** (p. 1181).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3386), **activemq::transport::inactivity::InactivityMonitor** (p. 2067), **activemq::transport::tcp::TcpTransport** (p. 3867), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2987).

**6.841.3.2** `void activemq::transport::TransportFilter::fire ( const Pointer< Command > & command ) [protected]`

Notify the listener of the new incoming Command.

### Parameters

<i>command</i>	- the command to send to the listener
----------------	---------------------------------------

**6.841.3.3** `void activemq::transport::TransportFilter::fire ( const decaf::lang::Exception & ex ) [protected]`

Notify the listener of the thrown Exception.

#### Parameters

<i>ex</i>	- the exception to send to listeners
-----------	--------------------------------------

**6.841.3.4** `virtual std::string activemq::transport::TransportFilter::getRemoteAddress ( ) const [inline, virtual]`

#### Returns

the remote address for this connection

Implements **activemq::transport::Transport** (p. 3998).

**6.841.3.5** `virtual TransportListener* activemq::transport::TransportFilter::getTransportListener ( ) const [inline, virtual]`

Gets the observer of asynchronous exceptions from this transport.

#### Returns

The listener of transport events.

Implements **activemq::transport::Transport** (p. 3998).

**6.841.3.6** `virtual bool activemq::transport::TransportFilter::isClosed ( ) const [inline, virtual]`

Has the **Transport** (p. 3996) been shutdown and no longer usable.

#### Returns

true if the **Transport** (p. 3996)

Implements **activemq::transport::Transport** (p. 3998).

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 3868).

**6.841.3.7** `virtual bool activemq::transport::TransportFilter::isConnected ( ) const [inline, virtual]`

Is the **Transport** (p. 3996) Connected to its Broker.

**Returns**

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 3998).

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 3868).

**6.841.3.8** `virtual bool activemq::transport::TransportFilter::isFaultTolerant ( ) const`  
`[inline, virtual]`

Is this **Transport** (p. 3996) fault tolerant, meaning that it will reconnect to a broker on disconnect.

**Returns**

true if the **Transport** (p. 3996) is fault tolerant.

Implements **activemq::transport::Transport** (p. 3999).

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 3868).

References `activemq::transport::Transport::isFaultTolerant()`.

**6.841.3.9** `virtual Transport* activemq::transport::TransportFilter::narrow ( const`  
`std::type_info & typeId ) [virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 3996) to allow a higher level transport to skip intermediate Transports in certain circumstances.

**Parameters**

<i>typeId</i>	- The type_info of the Object we are searching for.
---------------	---

**Returns**

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 3999).

**6.841.3.10** `virtual void activemq::transport::TransportFilter::onCommand ( const Pointer<`  
`Command > & command ) [virtual]`

Event handler for the receipt of a command.

**Parameters**

<i>command</i>	- the received command object.
----------------	--------------------------------

Implements **activemq::transport::TransportListener** (p. 4014).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3386), **activemq::transport::inactivity::InactivityMonitor** (p. 2067), **activemq::transport::logging::LoggingTransport** (p. 2478), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2987).

**6.841.3.11** `virtual void activemq::transport::TransportFilter::oneway ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )`  
[inline, virtual]

Sends a one-way command.

Does not wait for any response from the broker.

#### Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

#### Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 3999).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3386), **activemq::transport::inactivity::InactivityMonitor** (p. 2068), **activemq::transport::logging::LoggingTransport** (p. 2479), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2987).

**6.841.3.12** `virtual void activemq::transport::TransportFilter::onException ( const decaf::lang::Exception & ex )` [virtual]

Event handler for an exception from a command transport.

#### Parameters

<i>ex</i>	The exception to handle.
-----------	--------------------------

Implements **activemq::transport::TransportListener** (p. 4015).

Reimplemented in **activemq::transport::inactivity::InactivityMonitor** (p. 2068).

**6.841.3.13** `virtual void activemq::transport::TransportFilter::reconnect ( const decaf::net::URI & uri ) throw ( decaf::io::IOException )` [virtual]

reconnect to another location

#### Parameters

<i>uri</i>	
------------	--

## Exceptions

<i>IOException</i>	on failure of if not supported
--------------------	--------------------------------

Implements **activemq::transport::Transport** (p. 4000).

**6.841.3.14** `virtual Pointer<Response> activemq::transport::TransportFilter::request ( const Pointer< Command > & command, unsigned int timeout ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )`  
[inline, virtual]

Not supported by this class - throws an exception.

## Parameters

<i>command</i>	- The command that is sent as a request
<i>timeout</i>	- The the time to wait for a response.

## Exceptions

<i>IOException</i>	
<i>UnsupportedOperation Exception.</i>	

Implements **activemq::transport::Transport** (p. 4001).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3387), **activemq::transport::logging::LoggingTransport** (p. 2479), and **activemq::wireformat::openwire::OpenWireFormatN** (p. 2989).

**6.841.3.15** `virtual Pointer<Response> activemq::transport::TransportFilter::request ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )`  
[inline, virtual]

Not supported by this class - throws an exception.

## Parameters

<i>command</i>	the command that is sent as a request
----------------	---------------------------------------

## Exceptions

<i>IOException</i>	
<i>UnsupportedOperation Exception.</i>	

Implements **activemq::transport::Transport** (p. 4000).

Reimplemented in `activemq::transport::correlator::ResponseCorrelator` (p. 3387), `activemq::transport::logging::LoggingTransport` (p. 2479), and `activemq::wireformat::openwire::OpenWireFormat` (p. 2988).

**6.841.3.16** `virtual void activemq::transport::TransportFilter::setTransportListener ( TransportListener * listener ) [inline, virtual]`

Sets the observer of asynchronous exceptions from this transport.

#### Parameters

<i>listener</i>	the listener of transport events.
-----------------	-----------------------------------

Implements `activemq::transport::Transport` (p. 4001).

**6.841.3.17** `virtual void activemq::transport::TransportFilter::setWireFormat ( const Pointer< wireformat::WireFormat > & wireFormat ) [inline, virtual]`

Sets the WireFormat instance to use.

#### Parameters

<i>wireFormat</i>	The WireFormat the object used to encode / decode commands.
-------------------	---

Implements `activemq::transport::Transport` (p. 4002).

**6.841.3.18** `virtual void activemq::transport::TransportFilter::start ( ) throw ( decaf::io::IOException ) [virtual]`

Starts this transport object and creates the thread for polling on the input stream for commands.

If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

#### Exceptions

<i>IOException</i>	if an error occurs or if this transport has already been closed.
--------------------	--

Implements `activemq::transport::Transport` (p. 4002).

Reimplemented in `activemq::transport::correlator::ResponseCorrelator` (p. 3388), and `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p. 2989).

**6.841.3.19** `virtual void activemq::transport::TransportFilter::stop ( ) throw ( decaf::io::IOException ) [virtual]`

Stops the `Transport` (p. 3996).



**Exceptions**

<i>IOException</i>	if an error occurs while stopping the <b>Transport</b> (p. 3996).
--------------------	---

Implements **activemq::transport::Transport** (p. 4002).

**6.841.3.20** `virtual void activemq::transport::TransportFilter::transportInterrupted ( )`  
[virtual]

The transport has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 4015).

**6.841.3.21** `virtual void activemq::transport::TransportFilter::transportResumed ( )`  
[virtual]

The transport has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 4015).

**6.841.4 Field Documentation**

**6.841.4.1** `TransportListener* activemq::transport::TransportFilter::listener`  
[protected]

Listener of this transport.

**6.841.4.2** `Pointer<Transport> activemq::transport::TransportFilter::next`  
[protected]

The transport that this filter wraps around.

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/TransportFilter.h`

**6.842 activemq::transport::TransportListener Class Reference**

A listener of asynchronous exceptions from a command transport object.

```
#include <src/main/activemq/transport/TransportListener.h>
```

Inheritance diagram for `activemq::transport::TransportListener`:

## Public Member Functions

- virtual `~TransportListener ()`
- virtual void `onCommand (const Pointer< Command > &command)=0`  
*Event handler for the receipt of a command.*
- virtual void `onException (const decaf::lang::Exception &ex)=0`  
*Event handler for an exception from a command transport.*
- virtual void `transportInterrupted ()=0`  
*The transport has suffered an interruption from which it hopes to recover.*
- virtual void `transportResumed ()=0`  
*The transport has resumed after an interruption.*

### 6.842.1 Detailed Description

A listener of asynchronous exceptions from a command transport object.

### 6.842.2 Constructor & Destructor Documentation

6.842.2.1 virtual `activemq::transport::TransportListener::~~TransportListener ( )`  
`[inline, virtual]`

### 6.842.3 Member Function Documentation

6.842.3.1 virtual void `activemq::transport::TransportListener::onCommand ( const Pointer< Command > & command )` `[pure virtual]`

Event handler for the receipt of a command.

The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3996) deletes the command upon receipt.

#### Parameters

<i>command</i>	the received command object.
----------------	------------------------------

Implemented in `activemq::core::ActiveMQConnection` (p. 274), `activemq::transport::correlator::Response` (p. 3386), `activemq::transport::failover::FailoverTransportListener` (p. 1946), `activemq::transport::inactivity::InactivityMonitor` (p. 2067), `activemq::transport::logging::LoggingTransport` (p. 2478), `activemq::transport::mock::InternalCommandListener` (p. 2193), `activemq::transport::TransportFilter` (p. 4009), and `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p. 2987).

**6.842.3.2** `virtual void activemq::transport::TransportListener::onException ( const decaf::lang::Exception & ex ) [pure virtual]`

Event handler for an exception from a command transport.

#### Parameters

<i>ex</i>	The exception being propagated to this listener to handle.
-----------	--

Implemented in **activemq::core::ActiveMQConnection** (p. 275), **activemq::transport::failover::BackupTransport** (p. 760), **activemq::transport::failover::FailoverTransportListener** (p. 1946), **activemq::transport::inactivity::InactivityMonitor** (p. 2068), and **activemq::transport::TransportFilter** (p. 4010).

**6.842.3.3** `virtual void activemq::transport::TransportListener::transportInterrupted ( ) [pure virtual]`

The transport has suffered an interruption from which it hopes to recover.

Implemented in **activemq::core::ActiveMQConnection** (p. 281), **activemq::transport::DefaultTransportListener** (p. 1758), **activemq::transport::failover::FailoverTransportListener** (p. 1947), and **activemq::transport::TransportFilter** (p. 4013).

**6.842.3.4** `virtual void activemq::transport::TransportListener::transportResumed ( ) [pure virtual]`

The transport has resumed after an interruption.

Implemented in **activemq::core::ActiveMQConnection** (p. 281), **activemq::transport::DefaultTransportListener** (p. 1758), **activemq::transport::failover::FailoverTransportListener** (p. 1947), and **activemq::transport::TransportFilter** (p. 4013).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/TransportListener.h`

## 6.843 activemq::transport::TransportRegistry Class Reference

Registry of all **Transport** (p. 3996) Factories that are available to the client at runtime.

```
#include <src/main/activemq/transport/TransportRegistry.h>
```

### Public Member Functions

- `virtual ~TransportRegistry ()`
- `TransportFactory * findFactory (const std::string &name) const throw ( decaf::lang::exceptions::NoSuchElementException )`

*Gets a Registered **TransportFactory** (p. 4003) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.*

- void **registerFactory** (const std::string &name, **TransportFactory** \*factory) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException )

*Registers a new **TransportFactory** (p. 4003) with this Registry.*

- void **unregisterFactory** (const std::string &name)

*Unregisters the Factory with the given name and deletes that instance of the Factory.*

- std::vector< std::string > **getTransportNames** () const

*Retrieves a list of the names of all the Registered Transport's in this Registry.*

## Static Public Member Functions

- static **TransportRegistry** & **getInstance** ()

*Gets the single instance of the **TransportRegistry** (p. 4015).*

## 6.843.1 Detailed Description

Registry of all **Transport** (p. 3996) Factories that are available to the client at runtime. New Transport's must have a factory registered here before a connection attempt is made.

Since

3.0

## 6.843.2 Constructor & Destructor Documentation

- 6.843.2.1 virtual activemq::transport::TransportRegistry::~TransportRegistry ( )  
[virtual]

## 6.843.3 Member Function Documentation

- 6.843.3.1 **TransportFactory\*** activemq::transport::TransportRegistry::findFactory  
( const std::string & name ) const throw ( de-  
caf::lang::exceptions::NoSuchElementException  
)

Gets a Registered **TransportFactory** (p. 4003) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.

Parameters

<i>name</i>	The name of the Factory to find in the Registry.
-------------	--

**Returns**

the Factory registered under the given name.

**Exceptions**

<i>NoSuchElementException</i>	if no factory is registered with that name.
-------------------------------	---

**6.843.3.2** `static TransportRegistry& activemq::transport::TransportRegistry::getInstance ( ) [static]`

Gets the single instance of the **TransportRegistry** (p. 4015).

**Returns**

reference to the single instance of this Registry

**6.843.3.3** `std::vector<std::string> activemq::transport::TransportRegistry::getTransportNames ( ) const`

Retrieves a list of the names of all the Registered Transport's in this Registry.

**Returns**

stl vector of strings with all the **Transport** (p. 3996) names registered.

**6.843.3.4** `void activemq::transport::TransportRegistry::registerFactory ( const std::string & name, TransportFactory * factory ) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException )`

Registers a new **TransportFactory** (p. 4003) with this Registry.

If a Factory with the given name is already registered it is overwritten with the new one. Once a factory is added to the Registry its lifetime is controlled by the Registry, it will be deleted once the Registry has been deleted.

**Parameters**

<i>name</i>	The name of the new Factory to register.
<i>factory</i>	The new Factory to add to the Registry.

**Exceptions**

<i>IllegalArgumentException</i>	is name is the empty string.
<i>NullPointerException</i>	if the Factory is Null.

**6.843.3.5** void activemq::transport::TransportRegistry::unregisterFactory ( const std::string & name )

Unregisters the Factory with the given name and deletes that instance of the Factory.

**Parameters**

<i>name</i>	Name of the Factory to unregister and destroy
-------------	---

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**TransportRegistry.h**

**6.844 tree\_desc\_s Struct Reference**

```
#include <src/main/decaf/internal/util/zip/deflate.h>
```

**Data Fields**

- ct\_data \* dyn\_tree
- int max\_code
- static\_tree\_desc \* stat\_desc

**6.844.1 Field Documentation**

**6.844.1.1** ct\_data\* tree\_desc\_s::dyn\_tree

**6.844.1.2** int tree\_desc\_s::max\_code

**6.844.1.3** static\_tree\_desc\* tree\_desc\_s::stat\_desc

The documentation for this struct was generated from the following file:

- src/main/decaf/internal/util/zip/**deflate.h**

## 6.845 decaf::lang::Thread::UncaughtExceptionHandler Class Reference

Interface for handlers invoked when a **Thread** (p. 3876) abruptly terminates due to an uncaught exception.

```
#include <src/main/decaf/lang/Thread.h>
```

### Public Member Functions

- virtual **~UncaughtExceptionHandler** ()
- virtual void **uncaughtException** (const **Thread** \*thread, const **Throwable** &error)=0 throw ()

*Method invoked when the given thread terminates due to the given uncaught exception.*

#### 6.845.1 Detailed Description

Interface for handlers invoked when a **Thread** (p. 3876) abruptly terminates due to an uncaught exception.

#### 6.845.2 Constructor & Destructor Documentation

6.845.2.1 virtual decaf::lang::Thread::UncaughtExceptionHandler::~~UncaughtExceptionHandler ( ) [inline, virtual]

#### 6.845.3 Member Function Documentation

6.845.3.1 virtual void decaf::lang::Thread::UncaughtExceptionHandler::uncaughtException ( const **Thread** \* thread, const **Throwable** & error ) throw () [pure virtual]

Method invoked when the given thread terminates due to the given uncaught exception.

This method is defined to indicate that it will not throw an exception, throwing and exception from this method will on most systems result in a segmentation fault.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Thread.h**

## 6.846 decaf::net::UnknownHostException Class Reference

```
#include <src/main/decaf/net/UnknownHostException.h>
```

Inheritance diagram for decaf::net::UnknownHostException:

## Public Member Functions

- **UnknownHostException** () throw ()  
*Default Constructor.*
- **UnknownHostException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **UnknownHostException** (const **UnknownHostException** &ex) throw ()  
*Copy Constructor.*
- **UnknownHostException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **UnknownHostException** (const std::exception \*cause) throw ()  
*Constructor.*
- **UnknownHostException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **UnknownHostException** \* clone () const  
*Clones this exception.*
- virtual ~**UnknownHostException** () throw ()

## 6.846.1 Constructor & Destructor Documentation

6.846.1.1 decaf::net::UnknownHostException::UnknownHostException ( ) throw ()  
[inline]

Default Constructor.

6.846.1.2 decaf::net::UnknownHostException::UnknownHostException ( const Exception & ex )  
throw () [inline]

Conversion Constructor from some other Exception.

### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--



**6.846.1.3** decaf::net::UnknownHostException::UnknownHostException ( const  
UnknownHostException & ex ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.846.1.4** decaf::net::UnknownHostException::UnknownHostException ( const char \* *file*,  
const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ... ) throw ()  
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.846.1.5** decaf::net::UnknownHostException::UnknownHostException ( const std::exception \*  
*cause* ) throw () [inline]

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.846.1.6** decaf::net::UnknownHostException::UnknownHostException ( const char \* *file*,  
const int *lineNumber*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.846.1.7 `virtual decaf::net::UnknownHostException::~~UnknownHostException ( ) throw ()`  
`[inline, virtual]`

## 6.846.2 Member Function Documentation

6.846.2.1 `virtual UnknownHostException* decaf::net::UnknownHostException::clone ( )`  
`const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 2212).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/UnknownHostException.h`

## 6.847 decaf::net::UnknownServiceException Class Reference

```
#include <src/main/decaf/net/UnknownServiceException.h>
```

Inheritance diagram for `decaf::net::UnknownServiceException`:

### Public Member Functions

- **UnknownServiceException** ( ) throw ()  
*Default Constructor.*
- **UnknownServiceException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **UnknownServiceException** (const UnknownServiceException &ex) throw ()  
*Copy Constructor.*
- **UnknownServiceException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **UnknownServiceException** (const std::exception \*cause) throw ()

*Constructor.*

- **UnknownServiceException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **UnknownServiceException \* clone** () const  
*Clones this exception.*
- virtual ~**UnknownServiceException** () throw ()

### 6.847.1 Constructor & Destructor Documentation

**6.847.1.1 decaf::net::UnknownServiceException::UnknownServiceException ( ) throw ()**  
[inline]

Default Constructor.

**6.847.1.2 decaf::net::UnknownServiceException::UnknownServiceException ( const Exception & ex ) throw ()** [inline]

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.847.1.3 decaf::net::UnknownServiceException::UnknownServiceException ( const UnknownServiceException & ex ) throw ()** [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.847.1.4 decaf::net::UnknownServiceException::UnknownServiceException ( const char \* file, const int lineNumber, const std::exception \* cause, const char \* msg, ... ) throw ()**  
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.847.1.5** `decaf::net::UnknownServiceException::UnknownServiceException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.847.1.6** `decaf::net::UnknownServiceException::UnknownServiceException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.847.1.7** `virtual decaf::net::UnknownServiceException::~UnknownServiceException ( ) throw () [inline, virtual]`

## 6.847.2 Member Function Documentation

**6.847.2.1** `virtual UnknownServiceException* decaf::net::UnknownServiceException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from `decaf::io::IOException` (p. 2212).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/UnknownServiceException.h`

## 6.848 `decaf::io::UnsupportedEncodingException` Class Reference

Thrown when the the Character Encoding is not supported.

```
#include <src/main/decaf/io/UnsupportedEncodingException.h>
```

Inheritance diagram for `decaf::io::UnsupportedEncodingException`:

### Public Member Functions

- **`UnsupportedEncodingException () throw ()`**  
*Default Constructor.*
- **`UnsupportedEncodingException (const lang::Exception &ex) throw ()`**  
*Copy Constructor.*
- **`UnsupportedEncodingException (const UnsupportedEncodingException &ex) throw ()`**  
*Copy Constructor.*
- **`UnsupportedEncodingException (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()`**  
*Constructor - Initializes the file name and line number where this message occurred.*
- **`UnsupportedEncodingException (const std::exception *cause) throw ()`**  
*Constructor.*
- **`UnsupportedEncodingException (const char *file, const int lineNumber, const char *msg,...) throw ()`**  
*Constructor.*
- **`virtual UnsupportedEncodingException * clone () const`**  
*Clones this exception.*
- **`virtual ~UnsupportedEncodingException () throw ()`**

### 6.848.1 Detailed Description

Thrown when the the Character Encoding is not supported.

Since

1.0

### 6.848.2 Constructor & Destructor Documentation

**6.848.2.1** `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException ( )  
throw ()` `[inline]`

Default Constructor.

**6.848.2.2** `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException ( const  
lang::Exception & ex ) throw ()` `[inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

**6.848.2.3** `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException ( const  
UnsupportedEncodingException & ex ) throw ()` `[inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

**6.848.2.4** `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException ( const  
char * file, const int lineNumber, const std::exception * cause, const char * msg, ...  
) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.848.2.5** `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.848.2.6** `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor.

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.848.2.7** `virtual decaf::io::UnsupportedEncodingException::~~UnsupportedEncodingException ( ) throw () [inline, virtual]`

### 6.848.3 Member Function Documentation

**6.848.3.1** `virtual UnsupportedEncodingException* decaf::io::UnsupportedEncodingException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

A new instance of an Exception object that is a copy of this instance.

Reimplemented from **decaf::io::IOException** (p. 2212).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/UnsupportedEncodingException.h`

## 6.849 decaf::lang::exceptions::UnsupportedOperationException Class Reference

```
#include <src/main/decaf/lang/exceptions/UnsupportedOperationException.h>
```

Inheritance diagram for decaf::lang::exceptions::UnsupportedOperationException:

### Public Member Functions

- **UnsupportedOperationException** () throw ()  
*Default Constructor.*
- **UnsupportedOperationException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1886).*
- **UnsupportedOperationException** (const **UnsupportedOperationException** &ex) throw ()  
*Copy Constructor.*
- **UnsupportedOperationException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **UnsupportedOperationException** (const std::exception \*cause) throw ()  
*Constructor.*
- **UnsupportedOperationException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **UnsupportedOperationException** \* **clone** () const  
*Clones this exception.*
- virtual ~**UnsupportedOperationException** () throw ()

### 6.849.1 Constructor & Destructor Documentation

6.849.1.1 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException ( ) throw () [inline]

Default Constructor.



## 6.849 decaf::lang::exceptions::UnsupportedOperationException Class Reference

6.849.1.2 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException  
( const Exception & ex ) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1886).

### Parameters

<i>ex</i>	An exception that should become this type of <b>Exception</b> (p. 1886)
-----------	---

6.849.1.3 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException  
( const UnsupportedOperationException & ex ) throw () [inline]

Copy Constructor.

### Parameters

<i>ex</i>	An exception that should become this type of <b>Exception</b> (p. 1886)
-----------	---

6.849.1.4 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException  
( const char \* file, const int lineNumber, const std::exception \* cause, const char \*  
msg, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.849.1.5 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException  
( const std::exception \* cause ) throw () [inline]

Constructor.

### Parameters

<i>cause</i>	<b>Pointer</b> (p. 3032) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

**6.849.1.6** `decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException`  
`( const char * file, const int lineNumber, const char * msg, ... ) throw ()`  
`[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.849.1.7** `virtual decaf::lang::exceptions::UnsupportedOperationException::~UnsupportedOperationException`  
`( ) throw () [inline, virtual]`

## 6.849.2 Member Function Documentation

**6.849.2.1** `virtual UnsupportedOperationException*`  
`decaf::lang::exceptions::UnsupportedOperationException::clone ( ) const`  
`[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

an new **Exception** (p. 1886) instance that is a copy of this one.

Reimplemented from `decaf::lang::Exception` (p. 1889).

Reimplemented in `decaf::nio::ReadOnlyBufferException` (p. 3263).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/UnsupportedOperationException.h`

## 6.850 cms::UnsupportedOperationException Class Reference

This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

```
#include <src/main/cms/UnsupportedOperationException.h>
```

Inheritance diagram for `cms::UnsupportedOperationException`:

## Public Member Functions

- **UnsupportedOperationException** () throw ()
- **UnsupportedOperationException** (const **UnsupportedOperationException** &ex) throw ()
- **UnsupportedOperationException** (const std::string &message, const std::exception \*cause) throw ()
- **UnsupportedOperationException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**UnsupportedOperationException** () throw ()

### 6.850.1 Detailed Description

This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

Since

2.0

### 6.850.2 Constructor & Destructor Documentation

- 6.850.2.1 **cms::UnsupportedOperationException::UnsupportedOperationException** ( ) throw ()
- 6.850.2.2 **cms::UnsupportedOperationException::UnsupportedOperationException** ( const **UnsupportedOperationException** & ex ) throw ()
- 6.850.2.3 **cms::UnsupportedOperationException::UnsupportedOperationException** ( const std::string & message, const std::exception \* cause ) throw ()
- 6.850.2.4 **cms::UnsupportedOperationException::UnsupportedOperationException** ( const std::string & message, const std::exception \* cause, const std::vector< std::pair< std::string, int > > & stackTrace ) throw ()
- 6.850.2.5 virtual **cms::UnsupportedOperationException::~~UnsupportedOperationException** ( ) throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**UnsupportedOperationException.h**

## 6.851 decaf::net::URI Class Reference

This class represents an instance of a **URI** (p. 4031) as defined by RFC 2396.

```
#include <src/main/decaf/net/URI.h>
```

Inheritance diagram for decaf::net::URI:

## Public Member Functions

- **URI** ()  
*Default Constructor; same as calling a Constructor with all fields empty.*
- **URI** (const **URI** &uri) throw ( URISyntaxException )  
*Constructs a **URI** (p. 4031) as a copy of another **URI** (p. 4031).*
- **URI** (const std::string &uri) throw ( URISyntaxException )  
*Constructs a **URI** (p. 4031) from the given string.*
- **URI** (const std::string &scheme, const std::string &ssp, const std::string &fragment) throw ( URISyntaxException )  
*Constructs a **URI** (p. 4031) from the given components.*
- **URI** (const std::string &scheme, const std::string &userInfo, const std::string &host, int port, const std::string &path, const std::string &query, const std::string &fragment) throw ( URISyntaxException )  
*Constructs a **URI** (p. 4031) from the given components.*
- **URI** (const std::string &scheme, const std::string &host, const std::string &path, const std::string &fragment) throw ( URISyntaxException )  
*Constructs a **URI** (p. 4031) from the given components.*
- **URI** (const std::string &scheme, const std::string &authority, const std::string &path, const std::string &query, const std::string &fragment) throw ( URISyntaxException )  
*Constructs a **URI** (p. 4031) from the given components.*
- virtual ~**URI** ()
- virtual int **compareTo** (const **URI** &value) const  
*Compares this object with the specified object for order.*
- virtual bool **equals** (const **URI** &value) const
- virtual bool **operator==** (const **URI** &value) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **URI** &value) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- std::string **getAuthority** () const

- std::string **getFragment** () const
- std::string **getHost** () const
- std::string **getPath** () const
- int **getPort** () const
- std::string **getQuery** () const
- std::string **getScheme** () const
- std::string **getUserInfo** () const
- std::string **getRawAuthority** () const
  - Returns the raw authority component of this **URI** (p. 4031).*
- std::string **getRawFragment** () const
  - Returns the raw fragment component of this **URI** (p. 4031).*
- std::string **getRawPath** () const
  - Returns the raw path component of this **URI** (p. 4031).*
- std::string **getRawQuery** () const
  - Returns the raw query component of this **URI** (p. 4031).*
- std::string **getRawSchemeSpecificPart** () const
  - Returns the raw scheme-specific part of this **URI** (p. 4031).*
- std::string **getSchemeSpecificPart** () const
  - Returns the decoded scheme-specific part of this **URI** (p. 4031).*
- std::string **getRawUserInfo** () const
  - Returns the raw user-information component of this **URI** (p. 4031).*
- bool **isAbsolute** () const
  - Tells whether or not this **URI** (p. 4031) is absolute.*
- bool **isOpaque** () const
  - Tells whether or not this **URI** (p. 4031) is opaque.*
- **URI normalize** () const
  - Normalizes this **URI**'s path.*
- **URI parseServerAuthority** () const throw ( URISyntaxException )
  - Attempts to parse this **URI**'s authority component, if defined, into user-information, host, and port components.*
- **URI relativize** (const **URI** &uri) const
  - Relativizes the given **URI** (p. 4031) against this **URI** (p. 4031).*
- **URI resolve** (const std::string &str) const throw ( lang::exceptions::IllegalArgumentException )

*Constructs a new **URI** (p. 4031) by parsing the given string and then resolving it against this **URI** (p. 4031).*

- **URI resolve** (const **URI** &uri) const

*Resolves the given **URI** (p. 4031) against this **URI** (p. 4031).*

- std::string **toString** () const

*Returns the content of this **URI** (p. 4031) as a string.*

- **URL toURL** () const throw ( MalformedURLException, lang::exceptions::IllegalArgumentException )

*Constructs a **URL** (p. 4072) from this **URI** (p. 4031).*

## Static Public Member Functions

- static **URI create** (const std::string uri) throw ( lang::exceptions::IllegalArgumentException )

*Creates a **URI** (p. 4031) by parsing the given string.*

## 6.851.1 Detailed Description

This class represents an instance of a **URI** (p. 4031) as defined by RFC 2396.

## 6.851.2 Constructor & Destructor Documentation

### 6.851.2.1 decaf::net::URI::URI ( )

Default Constructor, same as calling a Constructor with all fields empty.

### 6.851.2.2 decaf::net::URI::URI ( const **URI** & uri ) throw ( URISyntaxException )

Constructs a **URI** (p. 4031) as a copy of another **URI** (p. 4031).

## Parameters

<i>uri</i>	- uri to copy
------------	---------------

### 6.851.2.3 decaf::net::URI::URI ( const std::string & uri ) throw ( URISyntaxException )

Constructs a **URI** (p. 4031) from the given string.

**Parameters**

<i>uri</i>	- string uri to parse.
------------	------------------------

**6.851.2.4** decaf::net::URI::URI ( const std::string & *scheme*, const std::string & *ssp*, const std::string & *fragment* ) throw ( URISyntaxException )

Constructs a **URI** (p. 4031) from the given components.

**Parameters**

<i>scheme</i>	- the uri scheme
<i>ssp</i>	- Scheme specific part
<i>fragment</i>	- Fragment

**6.851.2.5** decaf::net::URI::URI ( const std::string & *scheme*, const std::string & *userInfo*, const std::string & *host*, int *port*, const std::string & *path*, const std::string & *query*, const std::string & *fragment* ) throw ( URISyntaxException )

Constructs a **URI** (p. 4031) from the given components.

**Parameters**

<i>scheme</i>	- Scheme name
<i>userInfo</i>	- User name and authorization information
<i>host</i>	- Host name
<i>port</i>	- Port number
<i>path</i>	- Path
<i>query</i>	- Query
<i>fragment</i>	- Fragment

**6.851.2.6** decaf::net::URI::URI ( const std::string & *scheme*, const std::string & *host*, const std::string & *path*, const std::string & *fragment* ) throw ( URISyntaxException )

Constructs a **URI** (p. 4031) from the given components.

**Parameters**

<i>scheme</i>	- Scheme name
<i>host</i>	- Host name
<i>path</i>	- Path
<i>fragment</i>	- Fragment

**6.851.2.7** `decaf::net::URI::URI ( const std::string & scheme, const std::string & authority, const std::string & path, const std::string & query, const std::string & fragment ) throw ( URISyntaxException )`

Constructs a **URI** (p. 4031) from the given components.

#### Parameters

<i>scheme</i>	- Scheme name
<i>authority</i>	- Authority
<i>path</i>	- Path
<i>query</i>	- Query
<i>fragment</i>	- Fragment

**6.851.2.8** `virtual decaf::net::URI::~~URI ( ) [inline, virtual]`

### 6.851.3 Member Function Documentation

**6.851.3.1** `virtual int decaf::net::URI::compareTo ( const URI & value ) const [virtual]`

Compares this object with the specified object for order.

Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

#### Parameters

<i>value</i>	- the value to compare to this one.
--------------	-------------------------------------

#### Returns

zero if equal minus one if less than and one if greater than.

**6.851.3.2** `static URI decaf::net::URI::create ( const std::string uri ) throw ( lang::exceptions::IllegalArgumentException ) [static]`

Creates a **URI** (p. 4031) by parsing the given string.

This convenience factory method works as if by invoking the `URI(string)` constructor; any **URISyntaxException** (p. 4060) thrown by the constructor is caught and wrapped in a new `IllegalArgumentException` object, which is then thrown.

#### Parameters

<i>uri</i>	- <b>URI</b> (p. 4031) string to parse
------------	--

#### Exceptions

<i>IllegalArgumentException</i>	
---------------------------------	--



**6.851.3.3** virtual bool decaf::net::URI::equals ( const URI & *value* ) const [virtual]

**Returns**

true if this value is considered equal to the passed value.

**6.851.3.4** std::string decaf::net::URI::getAuthority ( ) const

**Returns**

the decoded authority component of this **URI** (p. 4031).

**6.851.3.5** std::string decaf::net::URI::getFragment ( ) const

**Returns**

the decoded fragment component of this **URI** (p. 4031).

**6.851.3.6** std::string decaf::net::URI::getHost ( ) const

**Returns**

the host component of this **URI** (p. 4031).

**6.851.3.7** std::string decaf::net::URI::getPath ( ) const

**Returns**

the path component of this **URI** (p. 4031).

**6.851.3.8** int decaf::net::URI::getPort ( ) const

**Returns**

the port component of this **URI** (p. 4031).

**6.851.3.9** std::string decaf::net::URI::getQuery ( ) const

**Returns**

the query component of this **URI** (p. 4031).

**6.851.3.10** `std::string decaf::net::URI::getRawAuthority ( ) const`

Returns the raw authority component of this **URI** (p. 4031).

The authority component of a **URI** (p. 4031), if defined, only contains the commercial-at character ('@') and characters in the unreserved, punct, escaped, and other categories. If the authority is server-based then it is further constrained to have valid user-information, host, and port components.

**Returns**

the raw authority component of the **URI** (p. 4031)

**6.851.3.11** `std::string decaf::net::URI::getRawFragment ( ) const`

Returns the raw fragment component of this **URI** (p. 4031).

The fragment component of a **URI** (p. 4031), if defined, only contains legal **URI** (p. 4031) characters.

**Returns**

the raw fragment component of this **URI** (p. 4031)

**6.851.3.12** `std::string decaf::net::URI::getRawPath ( ) const`

Returns the raw path component of this **URI** (p. 4031).

The path component of a **URI** (p. 4031), if defined, only contains the slash character ('/'), the commercial-at character ('@'), and characters in the unreserved, punct, escaped, and other categories.

**Returns**

the raw path component of this **URI** (p. 4031)

**6.851.3.13** `std::string decaf::net::URI::getRawQuery ( ) const`

Returns the raw query component of this **URI** (p. 4031).

The query component of a **URI** (p. 4031), if defined, only contains legal **URI** (p. 4031) characters.

**Returns**

the raw query component of the **URI** (p. 4031).

**6.851.3.14** `std::string decaf::net::URI::getRawSchemeSpecificPart ( ) const`

Returns the raw scheme-specific part of this **URI** (p. 4031).

The scheme-specific part is never undefined, though it may be empty. The scheme-specific part of a **URI** (p. 4031) only contains legal **URI** (p. 4031) characters.

**Returns**

the raw scheme special part of the uri

**6.851.3.15** `std::string decaf::net::URI::getRawUserInfo ( ) const`

Returns the raw user-information component of this **URI** (p. 4031).

The user-information component of a **URI** (p. 4031), if defined, only contains characters in the unreserved, punct, escaped, and other categories.

**Returns**

the raw user-information component of the **URI** (p. 4031)

**6.851.3.16** `std::string decaf::net::URI::getScheme ( ) const`**Returns**

the scheme component of this **URI** (p. 4031)

**6.851.3.17** `std::string decaf::net::URI::getSchemeSpecificPart ( ) const`

Returns the decoded scheme-specific part of this **URI** (p. 4031).

The string returned by this method is equal to that returned by the `getRawSchemeSpecificPart` method except that all sequences of escaped octets are decoded.

**Returns**

the raw scheme specific part of the uri.

**6.851.3.18** `std::string decaf::net::URI::getUserInfo ( ) const`**Returns**

the user info component of this **URI** (p. 4031)

**6.851.3.19 bool decaf::net::URI::isAbsolute ( ) const**

Tells whether or not this **URI** (p. 4031) is absolute.

A **URI** (p. 4031) is absolute if, and only if, it has a scheme component.

**Returns**

true if, and only if, this **URI** (p. 4031) is absolute

**6.851.3.20 bool decaf::net::URI::isOpaque ( ) const**

Tells whether or not this **URI** (p. 4031) is opaque.

A **URI** (p. 4031) is opaque if, and only if, it is absolute and its scheme-specific part does not begin with a slash character ('/'). An opaque **URI** (p. 4031) has a scheme, a scheme-specific part, and possibly a fragment; all other components are undefined.

**Returns**

true if, and only if, this **URI** (p. 4031) is opaque

**6.851.3.21 URI decaf::net::URI::normalize ( ) const**

Normalizes this **URI**'s path.

If this **URI** (p. 4031) is opaque, or if its path is already in normal form, then this **URI** (p. 4031) is returned. Otherwise a new **URI** (p. 4031) is constructed that is identical to this **URI** (p. 4031) except that its path is computed by normalizing this **URI**'s path in a manner consistent with RFC 2396, section 5.2, step 6, sub-steps c through f; that is:

1. All "." segments are removed. 2. If a ".." segment is preceded by a non-".." segment then both of these segments are removed. This step is repeated until it is no longer applicable. 3. If the path is relative, and if its first segment contains a colon character (':'), then a "." segment is prepended. This prevents a relative **URI** (p. 4031) with a path such as "a:b/c/d" from later being re-parsed as an opaque **URI** (p. 4031) with a scheme of "a" and a scheme-specific part of "b/c/d". (Deviation from RFC 2396)

A normalized path will begin with one or more "." segments if there were insufficient non-".." segments preceding them to allow their removal. A normalized path will begin with a "." segment if one was inserted by step 3 above. Otherwise, a normalized path will not contain any "." or ".." segments.

**Returns**

A **URI** (p. 4031) equivalent to this **URI** (p. 4031), but whose path is in normal form

**6.851.3.22** `virtual bool decaf::net::URI::operator< ( const URI & value ) const`  
`[virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

#### Returns

true if this object is equal to the one passed.

**6.851.3.23** `virtual bool decaf::net::URI::operator== ( const URI & value ) const`  
`[virtual]`

Compares equality between this object and the one passed.

#### Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

#### Returns

true if this object is equal to the one passed.

**6.851.3.24** `URI decaf::net::URI::parseServerAuthority ( ) const throw (`  
`URISyntaxException)`

Attempts to parse this URI's authority component, if defined, into user-information, host, and port components.

If this URI's authority component has already been recognized as being server-based then it will already have been parsed into user-information, host, and port components. In this case, or if this **URI** (p. 4031) has no authority component, this method simply returns this **URI** (p. 4031).

Otherwise this method attempts once more to parse the authority component into user-information, host, and port components, and throws an exception describing why the authority component could not be parsed in that way.

#### Returns

A **URI** (p. 4031) whose authority field has been parsed as a server-based authority

#### Exceptions

<i><b>URISyntaxException</b></i> (p. 4060)	- If the authority component of this <b>URI</b> (p. 4031) is defined but cannot be parsed as a server-based authority.
---	--

**6.851.3.25 URI decaf::net::URI::relativize ( const URI & uri ) const**

Relativizes the given **URI** (p. 4031) against this **URI** (p. 4031).

The relativization of the given **URI** (p. 4031) against this **URI** (p. 4031) is computed as follows:

1. If either this **URI** (p. 4031) or the given **URI** (p. 4031) are opaque, or if the scheme and authority components of the two URIs are not identical, or if the path of this **URI** (p. 4031) is not a prefix of the path of the given **URI** (p. 4031), then the given **URI** (p. 4031) is returned.
2. Otherwise a new relative hierarchical **URI** (p. 4031) is constructed with query and fragment components taken from the given **URI** (p. 4031) and with a path component computed by removing this URI's path from the beginning of the given URI's path.

**Parameters**

<i>uri</i>	- The <b>URI</b> (p. 4031) to be relativized against this <b>URI</b> (p. 4031)
------------	--

**Returns**

The resulting **URI** (p. 4031)

**6.851.3.26 URI decaf::net::URI::resolve ( const std::string & str ) const throw ( lang::exceptions::IllegalArgumentException )**

Constructs a new **URI** (p. 4031) by parsing the given string and then resolving it against this **URI** (p. 4031).

This convenience method works as if invoking it were equivalent to evaluating the expression `resolve( URI::create( str ) )`.

**Parameters**

<i>str</i>	- The string to be parsed into a <b>URI</b> (p. 4031)
------------	---

**Returns**

The resulting **URI** (p. 4031)

**Exceptions**

<i>IllegalArgumentException</i>	- If the given string violates RFC 2396
---------------------------------	---

**6.851.3.27 URI decaf::net::URI::resolve ( const URI & uri ) const**

Resolves the given **URI** (p. 4031) against this **URI** (p. 4031).

If the given **URI** (p. 4031) is already absolute, or if this **URI** (p. 4031) is opaque, then a copy of the given **URI** (p. 4031) is returned.

If the given URI's fragment component is defined, its path component is empty, and its scheme, authority, and query components are undefined, then a **URI** (p. 4031) with the given fragment but with all other components equal to those of this **URI** (p. 4031) is returned. This allows a **URI** (p. 4031) representing a standalone fragment reference, such as "#foo", to be usefully resolved against a base **URI** (p. 4031).

Otherwise this method constructs a new hierarchical **URI** (p. 4031) in a manner consistent with RFC 2396, section 5.2; that is:

1. A new **URI** (p. 4031) is constructed with this URI's scheme and the given URI's query and fragment components.
2. If the given **URI** (p. 4031) has an authority component then the new URI's authority and path are taken from the given **URI** (p. 4031).
3. Otherwise the new URI's authority component is copied from this **URI** (p. 4031), and its path is computed as follows:

1. If the given URI's path is absolute then the new URI's path is taken from the given **URI** (p. 4031).
2. Otherwise the given URI's path is relative, and so the new URI's path is computed by resolving the path of the given **URI** (p. 4031) against the path of this **URI** (p. 4031). This is done by concatenating all but the last segment of this URI's path, if any, with the given URI's path and then normalizing the result as if by invoking the `normalize` method.

The result of this method is absolute if, and only if, either this **URI** (p. 4031) is absolute or the given **URI** (p. 4031) is absolute.

#### Parameters

<i>uri</i>	- The <b>URI</b> (p. 4031) to be resolved against this <b>URI</b> (p. 4031)
------------	---

#### Returns

The resulting **URI** (p. 4031)

#### 6.851.3.28 `std::string decaf::net::URI::toString ( ) const`

Returns the content of this **URI** (p. 4031) as a string.

If this **URI** (p. 4031) was created by invoking one of the constructors in this class then a string equivalent to the original input string, or to the string computed from the originally-given components, as appropriate, is returned. Otherwise this **URI** (p. 4031) was created by normalization, resolution, or relativization, and so a string is constructed from this URI's components according to the rules specified in RFC 2396, section 5.2, step 7.

#### Returns

the string form of this **URI** (p. 4031)

#### 6.851.3.29 `URL decaf::net::URI::toURL ( ) const throw ( MalformedURLException, lang::exceptions::IllegalArgumentException )`

Constructs a **URL** (p. 4072) from this **URI** (p. 4031).

This convenience method works as if invoking it were equivalent to evaluating the expression `new URL (p. 4072)(this.toString())` after first checking that this **URI** (p. 4031) is absolute.

### Returns

A **URL** (p. 4072) constructed from this **URI** (p. 4031)

### Exceptions

<i><b>IllegalArgumentEx- ception</b></i>	- If this <b>URL</b> (p. 4072) is not absolute
<i><b>MalformedURLEx- ception</b></i> (p. 2536)	- If a protocol handler for the <b>URL</b> (p. 4072) could not be found, or if some other error occurred while constructing the <b>URL</b> (p. 4072)

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URL.h`

## 6.852 decaf::internal::net::URLEncoderDecoder Class Reference

```
#include <src/main/decaf/internal/net/URLEncoderDecoder.h>
```

### Public Member Functions

- **URLEncoderDecoder** ()
- virtual **~URLEncoderDecoder** ()

### Static Public Member Functions

- static void **validate** (const std::string &s, const std::string &legal) throw ( decaf::net::URISyntaxException )  
*Validate a string by checking if it contains any characters other than:*
- static void **validateSimple** (const std::string &s, const std::string &legal) throw ( decaf::net::URISyntaxException )  
*Validate a string by checking if it contains any characters other than:*
- static std::string **quoteIllegal** (const std::string &s, const std::string &legal)  
*All characters except letters ('a'..'z', 'A'..'Z') and numbers ('0'..'9') and legal characters are converted into their hexadecimal value prepended by ".*
- static std::string **encodeOthers** (const std::string &s)  
*Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars are not preserved.*



- static std::string **decode** (const std::string &s)

*Decodes the string argument which is assumed to be encoded in the x-www-form-urlencoded MIME content type using the UTF-8 encoding scheme.*

## 6.852.1 Constructor & Destructor Documentation

6.852.1.1 **decaf::internal::net::URLEncoderDecoder::URLEncoderDecoder ( )**

6.852.1.2 **virtual decaf::internal::net::URLEncoderDecoder::~~URLEncoderDecoder ( )**  
[inline, virtual]

## 6.852.2 Member Function Documentation

6.852.2.1 **static std::string decaf::internal::net::URLEncoderDecoder::decode ( const std::string & s )** [static]

Decodes the string argument which is assumed to be encoded in the x-www-form-urlencoded MIME content type using the UTF-8 encoding scheme.

” and two following hex digit characters are converted to the equivalent byte value. All other characters are passed through unmodified.

e.g. "A%20B%20C %24%25" -> "A B C \$%"

### Parameters

<i>s</i>	- The encoded string.
----------	-----------------------

### Returns

The decoded version.

6.852.2.2 **static std::string decaf::internal::net::URLEncoderDecoder::encodeOthers ( const std::string & s )** [static]

Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars are not preserved.

They are converted into their hexadecimal value prepended by ”.

For example: Euro currency symbol -> "%E2%82%AC".

### Parameters

<i>s</i>	- the string to be converted
----------	------------------------------

### Returns

the converted string

**6.852.2.3** `static std::string decaf::internal::net::URIEncoderDecoder::quotelllegal ( const std::string & s, const std::string & legal ) [static]`

All characters except letters ('a'..'z', 'A'..'Z') and numbers ('0'..'9') and legal characters are converted into their hexadecimal value prepended by ".

For example: '#' -> 23

Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars, are preserved.

#### Parameters

<i>s</i>	- the string to be converted
<i>legal</i>	- the characters allowed to be preserved in the string s

#### Returns

converted string

**6.852.2.4** `static void decaf::internal::net::URIEncoderDecoder::validate ( const std::string & s, const std::string & legal ) throw ( decaf::net::URISyntaxException ) [static]`

Validate a string by checking if it contains any characters other than:

1. letters ('a'..'z', 'A'..'Z')
2. numbers ('0'..'9')
3. characters in the legalset parameter
4. characters that are not ISO Control or are not ISO Space characters)

#### Parameters

<i>s</i>	- the string to be validated
<i>legal</i>	- the characters allowed in the string s

**6.852.2.5** `static void decaf::internal::net::URIEncoderDecoder::validateSimple ( const std::string & s, const std::string & legal ) throw ( decaf::net::URISyntaxException ) [static]`

Validate a string by checking if it contains any characters other than:

1. letters ('a'..'z', 'A'..'Z')
2. numbers ('0'..'9')
3. characters in the legalset parameter

#### Parameters

<i>s</i>	- the string to be validated
<i>legal</i>	- the characters allowed in the string s

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**URIEncoderDecoder.h**

## 6.853 decaf::internal::net::URIHelper Class Reference

Helper class used by the URI classes in encoding and decoding of URI's.

```
#include <src/main/decaf/internal/net/URIHelper.h>
```

### Public Member Functions

- **URIHelper** (const std::string &unreserved, const std::string &punct, const std::string &reserved, const std::string &someLegal, const std::string &allLegal)  
*Setup the **URIHelper** (p. 4047) with values assigned to the various fields that are used in the validation process.*
- **URIHelper** ()  
*Sets up the filter strings with sane defaults.*
- virtual ~**URIHelper** ()
- **URIType parseURI** (const std::string &uri, bool forceServer) throw ( decaf::net::URISyntaxException )  
*Parse the passed in URI.*
- void **validateScheme** (const std::string &uri, const std::string &scheme, int index) throw ( decaf::net::URISyntaxException )  
*Validate the schema portin of the URI.*
- void **validateSsp** (const std::string &uri, const std::string &ssp, std::size\_t index) throw ( decaf::net::URISyntaxException )  
*Validate that the URI Ssp Segment contains no invalid encodings.*
- void **validateAuthority** (const std::string &uri, const std::string &authority, std::size\_t index) throw ( decaf::net::URISyntaxException )  
*Validate that the URI Authority Segment contains no invalid encodings.*
- void **validatePath** (const std::string &uri, const std::string &path, std::size\_t index) throw ( decaf::net::URISyntaxException )  
*Validate that the URI Path Segment contains no invalid encodings.*
- void **validateQuery** (const std::string &uri, const std::string &query, std::size\_t index) throw ( decaf::net::URISyntaxException )  
*Validate that the URI Query Segment contains no invalid encodings.*
- void **validateFragment** (const std::string &uri, const std::string &fragment, std::size\_t index) throw ( decaf::net::URISyntaxException )  
*Validate that the URI fragment contains no invalid encodings.*
- **URIType parseAuthority** (bool forceServer, const std::string &authority) throw ( decaf::net::URISyntaxException )

*determine the host, port and user-info if the authority parses successfully to a server based authority*

- void **validateUserinfo** (const std::string &uri, const std::string &userinfo, std::size\_t index) throw ( decaf::net::URISyntaxException )  
*Check the supplied user info for validity.*
- bool **isValidHost** (bool forceServer, const std::string &host) throw ( decaf::net::URISyntaxException )  
*distinguish between IPv4, IPv6, domain name and validate it based on its type*
- bool **isValidDomainName** (const std::string &host)  
*Validates the string past to determine if it is a well formed domain name.*
- bool **isValidIPv4Address** (const std::string &host)  
*Validate if the host value is a well formed IPv4 address, this is the form XXX.XXX.XXX.XXX were X is any number 0-9.*
- bool **isValidIPv6Address** (const std::string &ipAddress)  
*Determines if the given address is valid according to the IPv6 spec.*
- bool **isValidIPv4Word** (const std::string &word)  
*Check is the string passed contains a Valid IPv4 word, which is an integer in the range of 0 to 255.*
- bool **isValidHexChar** (char c)  
*Determines if the given char is a valid Hex char.*

### 6.853.1 Detailed Description

Helper class used by the URI classes in encoding and decoding of URI's.

### 6.853.2 Constructor & Destructor Documentation

- 6.853.2.1 decaf::internal::net::URIHelper::URIHelper ( const std::string & *unreserved*, const std::string & *punct*, const std::string & *reserved*, const std::string & *someLegal*, const std::string & *allLegal* )

Setup the **URIHelper** (p. 4047) with values assigned to the various fields that are used in the validation process.

The defaults are overridden by these values.

#### Parameters

<i>unreserved</i>	- characters not reserved for use.
-------------------	------------------------------------

<i>punct</i>	- allowable punctuation symbols.
<i>reserved</i>	- characters not allowed for general use in the URI.
<i>someLegal</i>	- characters that are legal in certain cases.
<i>allLegal</i>	- characters that are always legal.

### 6.853.2.2 decaf::internal::net::URIHelper::URIHelper ( )

Sets up the filter strings with sane defaults.

### 6.853.2.3 virtual decaf::internal::net::URIHelper::~~URIHelper ( ) [inline, virtual]

## 6.853.3 Member Function Documentation

### 6.853.3.1 bool decaf::internal::net::URIHelper::isValidDomainName ( const std::string & *host* )

Validates the string past to determine if it is a well formed domain name.

#### Parameters

<i>host</i>	- domain name to validate.
-------------	----------------------------

#### Returns

true if host is well formed.

### 6.853.3.2 bool decaf::internal::net::URIHelper::isValidHexChar ( char *c* )

Determines if the given char is a valid Hex char.

Valid chars are A-F (upper or lower case) and 0-9.

#### Parameters

<i>c</i>	- char to inspect
----------	-------------------

#### Returns

true if *c* is a valid hex char.

### 6.853.3.3 bool decaf::internal::net::URIHelper::isValidHost ( bool *forceServer*, const std::string & *host* ) throw ( decaf::net::URISyntaxException )

distinguish between IPv4, IPv6, domain name and validate it based on its type

#### Parameters

<i>forceServer</i>	- true if the forceServer mode should be active.
<i>host</i>	- Host string to validate.

**Returns**

true if the host value if a valid domain name.

**Exceptions**

<i>URISyntaxException</i>	if the host is invalid and forceServer is true.
---------------------------	---

**6.853.3.4 bool decaf::internal::net::URIHelper::isValidIP4Word ( const std::string & word )**

Check is the string passed contains a Valid IPv4 word, which is an integer in the range of 0 to 255.

**Parameters**

<i>word</i>	- string value to check.
-------------	--------------------------

**Returns**

true if the word is a valid IPv4 word.

**6.853.3.5 bool decaf::internal::net::URIHelper::isValidIP6Address ( const std::string & ipAddress )**

Determines if the given address is valid according to the IPv6 spec.

**Parameters**

<i>ipAddress</i>	- string ip address value to validate.
------------------	--

**Returns**

true if the address string is valid.

**6.853.3.6 bool decaf::internal::net::URIHelper::isValidIPv4Address ( const std::string & host )**

Validate if the host value is a well formed IPv4 address, this is the form XXX.XXX.XXX.XXX were X is any number 0-9.

and XXX is not greater than 255.

**Parameters**

<i>host</i>	- IPv4 address string to parse.
-------------	---------------------------------

**Returns**

true if host is a well formed IPv4 address.

**6.853.3.7 URIType decaf::internal::net::URIHelper::parseAuthority ( bool *forceServer*, const std::string & *authority* ) throw ( decaf::net::URISyntaxException )**

determine the host, port and user-info if the authority parses successfully to a server based authority

behavior in error cases: if *forceServer* is true, throw URISyntaxException with the proper diagnostic messages. if *forceServer* is false assume this is a registry based uri, and just return leaving the host, port and user-info fields undefined.

and there are some error cases where URISyntaxException is thrown regardless of the *forceServer* parameter e.g. mal-formed ipv6 address

**Parameters**

<i>forceServer</i>	
<i>authority</i>	

**Returns**

a URIType (p. 4063) instance containing the parsed data.

**Exceptions**

URISyntaxException	
--------------------	--

**6.853.3.8 URIType decaf::internal::net::URIHelper::parseURI ( const std::string & *uri*, bool *forceServer* ) throw ( decaf::net::URISyntaxException )**

Parse the passed in URI.

**Parameters**

<i>uri</i>	- the URI to Parse
<i>forceServer</i>	- if true invalid URI data throws an Exception

**Returns**

a URIType (p. 4063) instance containing the parsed data.

**Exceptions**

URISyntaxException	if <i>forceServer</i> is true and the URI is invalid.
--------------------	---

**6.853.3.9** void decaf::internal::net::URIHelper::validateAuthority ( const std::string & *uri*, const std::string & *authority*, std::size\_t *index* ) throw ( decaf::net::URISyntaxException )

Validate that the URI Authority Segment contains no invalid encodings.

#### Parameters

<i>uri</i>	- the full uri.
<i>authority</i>	- the Authority to check.
<i>index</i>	- position in the uri where Authority starts.

#### Exceptions

<i>URISyntaxException</i>	if the fragment has errors.
---------------------------	-----------------------------

**6.853.3.10** void decaf::internal::net::URIHelper::validateFragment ( const std::string & *uri*, const std::string & *fragment*, std::size\_t *index* ) throw ( decaf::net::URISyntaxException )

Validate that the URI fragment contains no invalid encodings.

#### Parameters

<i>uri</i>	- the full uri.
<i>fragment</i>	- the fragment to check.
<i>index</i>	- position in the uri where fragment starts.

#### Exceptions

<i>URISyntaxException</i>	if the fragment has errors.
---------------------------	-----------------------------

**6.853.3.11** void decaf::internal::net::URIHelper::validatePath ( const std::string & *uri*, const std::string & *path*, std::size\_t *index* ) throw ( decaf::net::URISyntaxException )

Validate that the URI Path Segment contains no invalid encodings.

#### Parameters

<i>uri</i>	- the full uri.
<i>path</i>	- the path to check.
<i>index</i>	- position in the uri where path starts.

#### Exceptions

<i>URISyntaxException</i>	if the fragment has errors.
---------------------------	-----------------------------



**6.853.3.12** void decaf::internal::net::URIHelper::validateQuery ( const std::string & *uri*, const std::string & *query*, std::size\_t *index* ) throw ( decaf::net::URISyntaxException )

Validate that the URI Query Segment contains no invalid encodings.

#### Parameters

<i>uri</i>	- the full uri.
<i>query</i>	- the query to check.
<i>index</i>	- position in the uri where fragment starts.

#### Exceptions

<i>URISyntaxException</i>	if the fragment has errors.
---------------------------	-----------------------------

**6.853.3.13** void decaf::internal::net::URIHelper::validateScheme ( const std::string & *uri*, const std::string & *scheme*, int *index* ) throw ( decaf::net::URISyntaxException )

Validate the schema portin of the URI.

#### Parameters

<i>uri</i>	- the URI to check.
<i>scheme</i>	- the schema section of the URI.
<i>index</i>	- index in uri where schema starts.

#### Exceptions

<i>URISyntaxException</i>	if the fragment has errors.
---------------------------	-----------------------------

**6.853.3.14** void decaf::internal::net::URIHelper::validateSsp ( const std::string & *uri*, const std::string & *ssp*, std::size\_t *index* ) throw ( decaf::net::URISyntaxException )

Validate that the URI Ssp Segment contains no invalid encodings.

#### Parameters

<i>uri</i>	- the full uri.
<i>ssp</i>	- the SSP to check.
<i>index</i>	- position in the uri where Ssp starts.

#### Exceptions

<i>URISyntaxException</i>	if the fragment has errors.
---------------------------	-----------------------------

**6.853.3.15** `void decaf::internal::net::URIHelper::validateUserinfo ( const std::string  
& uri, const std::string & userinfo, std::size_t index ) throw (   
decaf::net::URISyntaxException )`

Check the supplied user info for validity.

#### Parameters

<i>uri</i>	- the uri to parse.
<i>userinfo</i>	- supplied user info
<i>index</i>	- index into the URI string where the data is located.

#### Returns

true if valid

#### Exceptions

<i>URISyntaxException</i>	if an error occurs
---------------------------	--------------------

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/URIHelper.h

## 6.854 activemq::transport::failover::URIPool Class Reference

```
#include <src/main/activemq/transport/failover/URIPool.h>
```

### Public Member Functions

- **URIPool ()**  
*Create an Empty URI Pool.*
- **URIPool (const decaf::util::List< URI > &uris)**  
*Creates a new URI Pool using the given list as the initial Free List.*
- **~URIPool ()**
- **URI getURI () throw ( decaf::lang::exceptions::NoSuchElementException )**  
*Fetches the next available URI from the pool, if there are no more URIs free when this method is called it throws a NoSuchElementException.*
- **void addURI (const URI &uri)**  
*Adds a URI to the free list, callers that have previously taken one using the getURI method should always return the URI when they close the resource that was connected to that URI.*
- **void addURIs (const StlList< URI > &uris)**

*Adds a List of URIs to this Pool, the method checks for duplicates already in the pool and does not add those.*

- void **removeURI** (const **URI** &uri)

*Remove a given URI from the Free List.*

- bool **isRandomize** () const

*Is the URI that is given randomly picked from the pool or is each one taken in sequence.*

- void **setRandomize** (bool value)

*Sets if the URI's that are taken from the pool are chosen Randomly or are taken in the order they are in the list.*

### 6.854.1 Constructor & Destructor Documentation

#### 6.854.1.1 activemq::transport::failover::URIPool::URIPool ( )

Create an Empty URI Pool.

#### 6.854.1.2 activemq::transport::failover::URIPool::URIPool ( const decaf::util::List< **URI** > & *uris* )

Creates a new URI Pool using the given list as the initial Free List.

##### Parameters

<i>uris</i>	- List of URI to place in the Pool.
-------------	-------------------------------------

#### 6.854.1.3 activemq::transport::failover::URIPool::~~URIPool ( )

### 6.854.2 Member Function Documentation

#### 6.854.2.1 void activemq::transport::failover::URIPool::addURI ( const **URI** & *uri* )

Adds a URI to the free list, callers that have previously taken one using the `getURI` method should always return the URI when they close the resource that was connected to that URI.

##### Parameters

<i>uri</i>	- a URI previously taken from the pool.
------------	---

**6.854.2.2** `void activemq::transport::failover::URIPool::addURIs ( const StlList< URI > & uris )`

Adds a List of URIs to this Pool, the method checks for duplicates already in the pool and does not add those.

#### Parameters

<i>uris</i>	- List of URIs to add into the Pool.
-------------	--------------------------------------

**6.854.2.3** `URI activemq::transport::failover::URIPool::getURI ( ) throw ( decaf::lang::exceptions::NoSuchElementException )`

Fetches the next available URI from the pool, if there are no more URIs free when this method is called it throws a NoSuchElementException.

Receiving the exception is not an indication that a URI won't be available in the future, the caller should react accordingly.

#### Returns

the next free URI in the Pool.

#### Exceptions

<i>NoSuchElementException</i>	if there are none free currently.
-------------------------------	-----------------------------------

**6.854.2.4** `bool activemq::transport::failover::URIPool::isRandomize ( ) const [inline]`

Is the URI that is given randomly picked from the pool or is each one taken in sequence.

#### Returns

true if URI gets are random.

**6.854.2.5** `void activemq::transport::failover::URIPool::removeURI ( const URI & uri )`

Remove a given URI from the Free List.

#### Parameters

<i>uri</i>	- the URI to find and remove from the free list
------------	---

**6.854.2.6** void **activemq::transport::failover::URIPool::setRandomize** ( bool *value* )  
[inline]

Sets if the URI's that are taken from the pool are chosen Randomly or are taken in the order they are in the list.

#### Parameters

<i>value</i>	- true indicates URI gets are random.
--------------	---------------------------------------

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/URIPool.h

## 6.855 activemq::util::URISupport Class Reference

```
#include <src/main/activemq/util/URISupport.h>
```

### Static Public Member Functions

- static void **parseURL** (const std::string &**URI**, **decaf::util::Properties** &properties) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Parses the properties out of the provided Broker URI and sets them in the passed Properties Object.*
- static **CompositeData** **parseComposite** (const **URI** &uri) throw ( decaf::net::URISyntaxException )  
*Parses a Composite URI into a Composite Data instance, the Composite URI takes the for scheme://(uri1,uri2,...uriN)?param1=value1, each of the composite URIs is stored in the CompositeData's internal list.*
- static **decaf::util::Properties** **parseQuery** (std::string query) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.*
- static void **parseQuery** (std::string query, **decaf::util::Properties** \*properties) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.*
- static std::string **createQueryString** (const **Properties** &options) throw ( decaf::net::URISyntaxException )  
*Given a properties object create a string that can be appended to a URI as a valid Query string.*

### 6.855.1 Member Function Documentation

**6.855.1.1** `static std::string activemq::util::URISupport::createQueryString ( const Properties & options ) throw ( decaf::net::URISyntaxException ) [static]`

Given a properties object create a string that can be appended to a URI as a valid Query string.

#### Parameters

<i>options</i>	Properties object containing key / value query values.
----------------	--

#### Returns

a valid URI query string.

#### Exceptions

<i>URISyntaxException</i>	if the string in the Properties object can't be encoded into a valid URI Query string.
---------------------------	--

**6.855.1.2** `static CompositeData activemq::util::URISupport::parseComposite ( const URI & uri ) throw ( decaf::net::URISyntaxException ) [static]`

Parses a Composite URI into a Composite Data instance, the Composite URI takes the form: `scheme://(uri1,uri2,...uriN)?param1=value1`, each of the composite URIs is stored in the CompositeData's internal list.

#### Parameters

<i>uri</i>	- The Composite URI to parse.
------------	-------------------------------

#### Returns

a new **CompositeData** (p. 1253) object with the parsed data

#### Exceptions

<i>URISyntaxException</i>	if the URI is not well formed.
---------------------------	--------------------------------

**6.855.1.3** `static void activemq::util::URISupport::parseQuery ( std::string query, decaf::util::Properties * properties ) throw ( decaf::lang::exceptions::IllegalArgumentException ) [static]`

Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

#### Parameters

<i>query</i>	- the query string to parse.
<i>properties</i>	- object pointer to get the parsed output.

### Exceptions

<i>IllegalArgumentException</i>	if the Query string is not well formed.
---------------------------------	---

**6.855.1.4** `static decaf::util::Properties activemq::util::URISupport::parseQuery ( std::string query ) throw ( decaf::lang::exceptions::IllegalArgumentException )`  
[static]

Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

### Parameters

<i>query</i>	The query string to parse and extract the encoded properties.
--------------	---

### Returns

Properties object with the parsed output.

### Exceptions

<i>IllegalArgumentException</i>	if the Query string is not well formed.
---------------------------------	---

**6.855.1.5** `static void activemq::util::URISupport::parseURL ( const std::string & URI, decaf::util::Properties & properties ) throw ( decaf::lang::exceptions::IllegalArgumentException )` [static]

Parses the properties out of the provided Broker URI and sets them in the passed Properties Object.

### Parameters

<i>URI</i>	a Broker URI to parse
<i>properties</i>	a Properties object to set the parsed values in

### Exceptions

<i>IllegalArgumentException</i>	if the passed URI is invalid
---------------------------------	------------------------------

The documentation for this class was generated from the following file:

- src/main/activemq/util/URISupport.h

## 6.856 decaf::net::URISyntaxException Class Reference

```
#include <src/main/decaf/net/URISyntaxException.h>
```

Inheritance diagram for decaf::net::URISyntaxException:

### Public Member Functions

- **URISyntaxException** () throw ()  
*Default Constructor.*
- **URISyntaxException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **URISyntaxException** (const **URISyntaxException** &ex) throw ()  
*Copy Constructor.*
- **URISyntaxException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **URISyntaxException** (const std::exception \*cause) throw ()  
*Constructor.*
- **URISyntaxException** (const char \*file, const int lineNumber, const char \*msg DECAF\_UNUSED) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **URISyntaxException** (const char \*file, const int lineNumber, const std::string &input, const std::string &reason) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **URISyntaxException** (const char \*file, const int lineNumber, const std::string &input, const std::string &reason, int index) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **URISyntaxException** \* clone () const  
*Clones this exception.*
- virtual ~**URISyntaxException** () throw ()
- std::string getInput () const
- std::string getReason () const
- int getIndex () const



### 6.856.1 Constructor & Destructor Documentation

**6.856.1.1** decaf::net::URISyntaxException::URISyntaxException ( ) throw () [inline]

Default Constructor.

**6.856.1.2** decaf::net::URISyntaxException::URISyntaxException ( const Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.856.1.3** decaf::net::URISyntaxException::URISyntaxException ( const URISyntaxException & ex ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.856.1.4** decaf::net::URISyntaxException::URISyntaxException ( const char \* file, const int lineNumber, const std::exception \* cause, const char \* msg, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.856.1.5** decaf::net::URISyntaxException::URISyntaxException ( const std::exception \* cause ) throw () [inline]

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.856.1.6** `decaf::net::URISyntaxException::URISyntaxException ( const char * file, const int lineNumber, const char *msg DECAF_UNUSED ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.  
Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.856.1.7** `decaf::net::URISyntaxException::URISyntaxException ( const char * file, const int lineNumber, const std::string & input, const std::string & reason ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.  
Sets the input string that caused the error and the reason for the error.

**Parameters**

<i>file</i>	The file name where exception occurs.
<i>lineNumber</i>	The line number where the exception occurred.
<i>input</i>	The <b>URL</b> (p. 4072) that caused the exception.
<i>reason</i>	The reason for the failure.

**6.856.1.8** `decaf::net::URISyntaxException::URISyntaxException ( const char * file, const int lineNumber, const std::string & input, const std::string & reason, int index ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.  
Sets the input string that caused the error and the reason for the error.

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>input</i>	The input <b>URI</b> (p. 4031) that caused the exception
<i>reason</i>	The reason for the failure.
<i>index</i>	The index in the <b>URI</b> (p. 4031) string where the error occurred.

6.856.1.9 `virtual decaf::net::URISyntaxException::~~URISyntaxException ( ) throw ()`  
`[inline, virtual]`

## 6.856.2 Member Function Documentation

6.856.2.1 `virtual URISyntaxException* decaf::net::URISyntaxException::clone ( ) const`  
`[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::lang::Exception** (p. 1889).

6.856.2.2 `int decaf::net::URISyntaxException::getIndex ( ) const` `[inline]`

### Returns

the index in the input string where the error occurred or -1

6.856.2.3 `std::string decaf::net::URISyntaxException::getInput ( ) const` `[inline]`

### Returns

the Input string that cause this exception or ""

6.856.2.4 `std::string decaf::net::URISyntaxException::getReason ( ) const` `[inline]`

### Returns

the Reason given for this failure, or ""

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URISyntaxException.h`

## 6.857 decaf::internal::net::URIType Class Reference

Basic type object that holds data that composes a given URI.

```
#include <src/main/decaf/internal/net/URIType.h>
```

## Public Member Functions

- **URIType** (const std::string &source)
- **URIType** ()
- virtual ~**URIType** ()
- std::string **getSource** () const  
*Gets the source URI string that was parsed to obtain this **URIType** (p. 4063) instance and the resulting data,.*
- void **setSource** (const std::string &source)  
*Sets the source URI string that was parsed to obtain this **URIType** (p. 4063) instance and the resulting data,.*
- std::string **getScheme** () const  
*Gets the Scheme of the URI, e.g.*
- void **setScheme** (const std::string &scheme)  
*Sets the Scheme of the URI, e.g.*
- std::string **getSchemeSpecificPart** () const  
*Gets the Scheme Specific Part of the URI.*
- void **setSchemeSpecificPart** (const std::string &schemeSpecificPart)  
*Sets the Scheme Specific Part of the URI.*
- std::string **getAuthority** () const  
*Gets the Authority of the URI.*
- void **setAuthority** (const std::string &authority)  
*Sets the Authority of the URI.*
- std::string **getUserInfo** () const  
*Gets the user info part of the URI, e.g.*
- void **setUserInfo** (const std::string &userinfo)  
*Sets the user info part of the URI, e.g.*
- std::string **getHost** () const  
*Gets the Host name part of the URI.*
- void **setHost** (const std::string &host)  
*Sets the Host name part of the URI.*
- int **getPort** () const  
*Gets the port part of the URI.*
- void **setPort** (int port)

*Sets the port part of the URI.*

- **std::string getPath () const**  
*Gets the Path part of the URI.*
- **void setPath (const std::string &path)**  
*Sets the Path part of the URI.*
- **std::string getQuery () const**  
*Gets the Query part of the URI.*
- **void setQuery (const std::string &query)**  
*Sets the Query part of the URI.*
- **std::string getFragment () const**  
*Gets the Fragment part of the URI.*
- **void setFragment (const std::string &fragment)**  
*Sets the Fragment part of the URI.*
- **bool isOpaque () const**  
*Gets if the URI is Opaque.*
- **void setOpaque (bool opaque)**  
*Sets if the URI is Opaque.*
- **bool isAbsolute () const**  
*Gets if the URI is Absolute.*
- **void setAbsolute (bool absolute)**  
*Sets if the URI is Absolute.*
- **bool isServerAuthority () const**  
*Gets if the URI is a Server Authority.*
- **void setServerAuthority (bool serverAuthority)**  
*Sets if the URI is a Server Authority.*
- **bool isValid () const**  
*Gets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.*
- **void setValid (bool valid)**  
*Sets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.*

### 6.857.1 Detailed Description

Basic type object that holds data that composes a given URI.

### 6.857.2 Constructor & Destructor Documentation

**6.857.2.1** `decaf::internal::net::URIType::URIType ( const std::string & source )` `[inline]`

**6.857.2.2** `decaf::internal::net::URIType::URIType ( )` `[inline]`

**6.857.2.3** `virtual decaf::internal::net::URIType::~~URIType ( )` `[inline, virtual]`

### 6.857.3 Member Function Documentation

**6.857.3.1** `std::string decaf::internal::net::URIType::getAuthority ( ) const` `[inline]`

Gets the Authority of the URI.

#### Returns

Authority part string.

**6.857.3.2** `std::string decaf::internal::net::URIType::getFragment ( ) const` `[inline]`

Gets the Fragment part of the URI.

#### Returns

Fragment part string.

**6.857.3.3** `std::string decaf::internal::net::URIType::getHost ( ) const` `[inline]`

Gets the Host name part of the URI.

#### Returns

Host name part string.

**6.857.3.4** `std::string decaf::internal::net::URIType::getPath ( ) const` `[inline]`

Gets the Path part of the URI.

#### Returns

Path part string.

**6.857.3.5** `int decaf::internal::net::URIType::getPort ( ) const [inline]`

Gets the port part of the URI.

**Returns**

port part string, -1 if not set.

**6.857.3.6** `std::string decaf::internal::net::URIType::getQuery ( ) const [inline]`

Gets the Query part of the URI.

**Returns**

Query part string.

**6.857.3.7** `std::string decaf::internal::net::URIType::getScheme ( ) const [inline]`

Gets the Scheme of the URI, e.g.

scheme ("http"/"ftp"/...).

**Returns**

scheme part string.

**6.857.3.8** `std::string decaf::internal::net::URIType::getSchemeSpecificPart ( ) const [inline]`

Gets the Scheme Specific Part of the URI.

**Returns**

scheme specific part string.

**6.857.3.9** `std::string decaf::internal::net::URIType::getSource ( ) const [inline]`

Gets the source URI string that was parsed to obtain this **URIType** (p.4063) instance and the resulting data,.

**Returns**

the source URI string

**6.857.3.10** `std::string decaf::internal::net::URIType::getUserInfo ( ) const` `[inline]`

Gets the user info part of the URI, e.g.

user name, as in `http://user:passwd@host:port/`

**Returns**

user info part string.

**6.857.3.11** `bool decaf::internal::net::URIType::isAbsolute ( ) const` `[inline]`

Gets if the URI is Absolute.

**Returns**

true if Absolute.

**6.857.3.12** `bool decaf::internal::net::URIType::isOpaque ( ) const` `[inline]`

Gets if the URI is Opaque.

**Returns**

true if opaque.

**6.857.3.13** `bool decaf::internal::net::URIType::isServerAuthority ( ) const` `[inline]`

Gets if the URI is a Server Authority.

**Returns**

true if Server Authority.

**6.857.3.14** `bool decaf::internal::net::URIType::isValid ( ) const` `[inline]`

Gets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

**Returns**

true if the **URIType** (p. 4063) contains valid data.

**6.857.3.15** `void decaf::internal::net::URIType::setAbsolute ( bool absolute )` `[inline]`

Sets if the URI is Absolute.

**Parameters**



<i>absolute</i>	- true if Absolute.
-----------------	---------------------

**6.857.3.16** void decaf::internal::net::URIType::setAuthority ( const std::string & *authority* )  
[inline]

Sets the Authority of the URI.

#### Parameters

<i>authority</i>	Authority part string.
------------------	------------------------

**6.857.3.17** void decaf::internal::net::URIType::setFragment ( const std::string & *fragment* )  
[inline]

Sets the Fragment part of the URI.

#### Parameters

<i>fragment</i>	- Fragment part string.
-----------------	-------------------------

**6.857.3.18** void decaf::internal::net::URIType::setHost ( const std::string & *host* )  
[inline]

Sets the Host name part of the URI.

#### Parameters

<i>host</i>	- Host name part string.
-------------	--------------------------

**6.857.3.19** void decaf::internal::net::URIType::setOpaque ( bool *opaque* ) [inline]

Sets if the URI is Opaque.

#### Parameters

<i>opaque</i>	true if opaque.
---------------	-----------------

**6.857.3.20** void decaf::internal::net::URIType::setPath ( const std::string & *path* )  
[inline]

Sets the Path part of the URI.

**Parameters**

<i>path</i>	- Path part string.
-------------	---------------------

**6.857.3.21** `void decaf::internal::net::URIType::setPort ( int port )` `[inline]`

Sets the port part of the URI.

**Parameters**

<i>port</i>	- port part string, -1 if not set.
-------------	------------------------------------

**6.857.3.22** `void decaf::internal::net::URIType::setQuery ( const std::string & query )`  
`[inline]`

Sets the Query part of the URI.

**Parameters**

<i>query</i>	- Query part string.
--------------	----------------------

**6.857.3.23** `void decaf::internal::net::URIType::setScheme ( const std::string & scheme )`  
`[inline]`

Sets the Scheme of the URI, e.g.

scheme ("http"/"ftp"/...).

**Parameters**

<i>scheme</i>	- scheme part string.
---------------	-----------------------

**6.857.3.24** `void decaf::internal::net::URIType::setSchemeSpecificPart ( const std::string & schemeSpecificPart )` `[inline]`

Sets the Scheme Specific Part of the URI.

**Parameters**

<i>scheme-SpecificPart</i>	- scheme specific part string.
----------------------------	--------------------------------

**6.857.3.25** void decaf::internal::net::URIType::setServerAuthority ( bool *serverAuthority* )  
[inline]

Sets if the URI is a Server Authority.

#### Parameters

<i>server-Authority</i>	- true if Server Authority.
-------------------------	-----------------------------

**6.857.3.26** void decaf::internal::net::URIType::setSource ( const std::string & *source* )  
[inline]

Sets the source URI string that was parsed to obtain this **URIType** (p. 4063) instance and the resulting data,.

#### Parameters

<i>source</i>	- the source URI string
---------------	-------------------------

**6.857.3.27** void decaf::internal::net::URIType::setUserInfo ( const std::string & *userinfo* )  
[inline]

Sets the user info part of the URI, e.g.

user name, as in `http://user:passwd@host:port/`

#### Parameters

<i>userinfo</i>	- user info part string.
-----------------	--------------------------

**6.857.3.28** void decaf::internal::net::URIType::setValid ( bool *valid* ) [inline]

Sets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

#### Parameters

<i>valid</i>	- true if the <b>URIType</b> (p. 4063) contains valid data.
--------------	---

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/URIType.h`

## 6.858 decaf::net::URL Class Reference

Class **URL** (p. 4072) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.

```
#include <src/main/decaf/net/URL.h>
```

### Public Member Functions

- **URL** ()
- **URL** (const std::string &url)
- virtual ~**URL** ()

#### 6.858.1 Detailed Description

Class **URL** (p. 4072) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web. A resource can be something as simple as a file or a directory, or it can be a reference to a more complicated object, such as a query to a database or to a search engine. More information on the types of URLs and their formats can be found at:

```
http://www.ksc.nasa.gov/facts/internet/url-primer.html
```

In general, a **URL** (p. 4072) can be broken into several parts. The previous example of a **URL** (p. 4072) indicates that the protocol to use is http (HyperText Transfer Protocol) and that the information resides on a host machine named `www.ksc.nasa.gov`. The information on that host machine is named `/facts/internet/url-primer.html`. The exact meaning of this name on the host machine is both protocol dependent and host dependent. The information normally resides in a file, but it could be generated on the fly. This component of the **URL** (p. 4072) is called the path component.

A **URL** (p. 4072) can optionally specify a "port", which is the port number to which the TCP connection is made on the remote host machine. If the port is not specified, the default port for the protocol is used instead. For example, the default port for http is 80. An alternative port could be specified as:

```
http://www.ksc.nasa.gov:80/facts/internet/url-primer.html
```

The syntax of **URL** (p. 4072) is defined by RFC 2396: Uniform Resource Identifiers (**URI** (p. 4031)): Generic Syntax, amended by RFC 2732: Format for Literal IPv6 Addresses in URLs. The Literal IPv6 address format also supports `scope_ids`. The syntax and usage of `scope_ids` is described here.

A **URL** (p. 4072) may have appended to it a "fragment", also known as a "ref" or a "reference". The fragment is indicated by the sharp sign character "#" followed by more characters. For example,

```
http://www.apache.org/cms/index.html#chapter1
```

This fragment is not technically part of the **URL** (p. 4072). Rather, it indicates that after the specified resource is retrieved, the application is specifically interested in that

part of the document that has the tag `chapter1` attached to it. The meaning of a tag is resource specific.

An application can also specify a "relative URL", which contains only enough information to reach the resource relative to another **URL** (p. 4072). Relative URLs are frequently used within HTML pages. For example, if the contents of the **URL** (p. 4072):

```
http://www.apache.org/cms/index.html
```

contained within it the relative **URL** (p. 4072):

```
FAQ.html
```

it would be a shorthand for:

```
http://www.apache.org/cms/FAQ.html
```

The relative **URL** (p. 4072) need not specify all the components of a **URL** (p. 4072). If the protocol, host name, or port number is missing, the value is inherited from the fully specified **URL** (p. 4072). The file component must be specified. The optional fragment is not inherited.

The **URL** (p. 4072) class does not itself encode or decode any **URL** (p. 4072) components according to the escaping mechanism defined in RFC2396. It is the responsibility of the caller to encode any fields, which need to be escaped prior to calling **URL** (p. 4072), and also to decode any escaped fields, that are returned from **URL** (p. 4072). Furthermore, because **URL** (p. 4072) has no knowledge of **URL** (p. 4072) escaping, it does not recognise equivalence between the encoded or decoded form of the same **URL** (p. 4072). For example, the two URLs:

```
http://foo.com/hello world/ and http://foo.com/hello%20world
```

would be considered not equal to each other.

Note, the **URI** (p. 4031) class does perform escaping of its component fields in certain circumstances. The recommended way to manage the encoding and decoding of URLs is to use **URI** (p. 4031), and to convert between these two classes using `toURI()` and `URLtoURL()` (p. 4043).

The **URLEncoder** (p. 4074) and **URLDecoder** (p. 4074) classes can also be used, but only for HTML form encoding, which is not the same as the encoding scheme defined in RFC2396.

## 6.858.2 Constructor & Destructor Documentation

**6.858.2.1** `decaf::net::URL::URL ( )`

**6.858.2.2** `decaf::net::URL::URL ( const std::string & url )`

**6.858.2.3** `virtual decaf::net::URL::~~URL ( ) [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URL.h`

## 6.859 decaf::net::URLDecoder Class Reference

```
#include <src/main/decaf/net/URLDecoder.h>
```

### Public Member Functions

- virtual `~URLDecoder()`

### Static Public Member Functions

- static `std::string decode(const std::string &value)`

*Decodes the string argument which is assumed to be encoded in the x-www-form-urlencoded MIME content type.*

### 6.859.1 Constructor & Destructor Documentation

6.859.1.1 virtual `decaf::net::URLDecoder::~URLDecoder()` [`inline`, `virtual`]

### 6.859.2 Member Function Documentation

6.859.2.1 static `std::string decaf::net::URLDecoder::decode(const std::string &value)`  
[`static`]

Decodes the string argument which is assumed to be encoded in the x-www-form-urlencoded MIME content type.

'+' will be converted to space, '%' and two following hex digit characters are converted to the equivalent byte value. All other characters are passed through unmodified.

e.g. "A+B+C %24%25" -> "A B C \$%"

#### Parameters

<i>value</i>	- string The encoded string.
--------------	------------------------------

#### Returns

The decoded version as a string.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URLDecoder.h`

## 6.860 decaf::net::URLEncoder Class Reference

```
#include <src/main/decaf/net/URLEncoder.h>
```

## Public Member Functions

- virtual `~URLEncoder ()`

## Static Public Member Functions

- static `std::string encode (const std::string &value)`

*This class contains a utility method for converting a string to the format required by the `application/x-www-form-urlencoded` MIME content type.*

### 6.860.1 Constructor & Destructor Documentation

6.860.1.1 `virtual decaf::net::URLEncoder::~URLEncoder ( ) [inline, virtual]`

### 6.860.2 Member Function Documentation

6.860.2.1 `static std::string decaf::net::URLEncoder::encode ( const std::string & value ) [static]`

This class contains a utility method for converting a string to the format required by the `application/x-www-form-urlencoded` MIME content type.

All characters except letters ('a'..'z', 'A'..'Z') and numbers ('0'..'9') and characters '.', '-', '\*', '\_' are converted into their hexadecimal value prepended by ".

For example: '#' -> 23

In addition, spaces are substituted by '+'

#### Parameters

<i>value</i>	- the string to be converted
--------------	------------------------------

#### Returns

the converted string

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URLEncoder.h`

## 6.861 activemq::util::Usage Class Reference

```
#include <src/main/activemq/util/Usage.h>
```

Inheritance diagram for `activemq::util::Usage`:

## Public Member Functions

- virtual `~Usage ()`
- virtual void `waitForSpace ()=0`  
*Waits forever for more space to be returned to this **Usage** (p. 4075) Manager.*
- virtual void `waitForSpace (unsigned int timeout)=0`  
*Waits for more space to be returned to this **Usage** (p. 4075) Manager, times out when the given time span in milliseconds elapses.*
- virtual void `enqueueUsage (unsigned long long value)=0`  
*Tries to increase the usage by value amount but blocks if this object is currently full.*
- virtual void `increaseUsage (unsigned long long value)=0`  
*Increases the usage by the value amount.*
- virtual void `decreaseUsage (unsigned long long value)=0`  
*Decreases the usage by the value amount.*
- virtual bool `isFull () const =0`  
*Returns true if this **Usage** (p. 4075) instance is full, i.e.*

## 6.861.1 Constructor & Destructor Documentation

6.861.1.1 virtual `activemq::util::Usage::~~Usage ( )` [`inline`, `virtual`]

## 6.861.2 Member Function Documentation

6.861.2.1 virtual void `activemq::util::Usage::decreaseUsage ( unsigned long long value )`  
`[pure virtual]`

Decreases the usage by the value amount.

### Parameters

<i>value</i>	Amount of space to return to the pool
--------------	---------------------------------------

Implemented in `activemq::util::MemoryUsage` (p. 2594).

6.861.2.2 virtual void `activemq::util::Usage::enqueueUsage ( unsigned long long value )`  
`[pure virtual]`

Tries to increase the usage by value amount but blocks if this object is currently full.

### Parameters

<i>value</i>	Amount of usage in bytes to add.
--------------	----------------------------------



Implemented in **activemq::util::MemoryUsage** (p. 2594).

**6.861.2.3** `virtual void activemq::util::Usage::increaseUsage ( unsigned long long value )`  
`[pure virtual]`

Increases the usage by the value amount.

#### Parameters

<i>value</i>	Amount of usage to add.
--------------	-------------------------

Implemented in **activemq::util::MemoryUsage** (p. 2595).

**6.861.2.4** `virtual bool activemq::util::Usage::isFull ( ) const` `[pure virtual]`

Returns true if this **Usage** (p. 4075) instance is full, i.e.

**Usage** (p. 4075)  $\geq 100\%$

#### Returns

true if **Usage** (p. 4075) is at the Full point.

Implemented in **activemq::util::MemoryUsage** (p. 2595).

**6.861.2.5** `virtual void activemq::util::Usage::waitForSpace ( unsigned int timeout )` `[pure virtual]`

Waits for more space to be returned to this **Usage** (p. 4075) Manager, times out when the given time span in milliseconds elapses.

#### Parameters

<i>timeout</i>	The time to wait for more space.
----------------	----------------------------------

Implemented in **activemq::util::MemoryUsage** (p. 2596).

**6.861.2.6** `virtual void activemq::util::Usage::waitForSpace ( )` `[pure virtual]`

Waits forever for more space to be returned to this **Usage** (p. 4075) Manager.

Implemented in **activemq::util::MemoryUsage** (p. 2595).

The documentation for this class was generated from the following file:

- `src/main/activemq/util/Usage.h`

## 6.862 decaf::io::UTFDataFormatException Class Reference

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

```
#include <src/main/decaf/io/UTFDataFormatException.h>
```

Inheritance diagram for decaf::io::UTFDataFormatException:

### Public Member Functions

- **UTFDataFormatException** () throw ()  
*Default Constructor.*
- **UTFDataFormatException** (const lang::Exception &ex) throw ()  
*Copy Constructor.*
- **UTFDataFormatException** (const UTFDataFormatException &ex) throw ()  
*Copy Constructor.*
- **UTFDataFormatException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **UTFDataFormatException** (const std::exception \*cause) throw ()  
*Constructor.*
- **UTFDataFormatException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **UTFDataFormatException** \* clone () const  
*Clones this exception.*
- virtual ~**UTFDataFormatException** () throw ()

### 6.862.1 Detailed Description

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

Since

1.0

## 6.862.2 Constructor & Destructor Documentation

**6.862.2.1** decaf::io::UTFDataFormatException::UTFDataFormatException ( ) throw ()  
[inline]

Default Constructor.

**6.862.2.2** decaf::io::UTFDataFormatException::UTFDataFormatException ( const lang::Exception & ex ) throw () [inline]

Copy Constructor.

### Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

**6.862.2.3** decaf::io::UTFDataFormatException::UTFDataFormatException ( const UTFDataFormatException & ex ) throw () [inline]

Copy Constructor.

### Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

**6.862.2.4** decaf::io::UTFDataFormatException::UTFDataFormatException ( const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ... ) throw ()  
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.862.2.5** decaf::io::UTFDataFormatException::UTFDataFormatException ( const std::exception \* *cause* ) throw () [inline]

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.862.2.6** `decaf::io::UTFDataFormatException::UTFDataFormatException ( const char * file,  
const int lineNumber, const char * msg, ... ) throw ()` `[inline]`

Constructor.

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.862.2.7** `virtual decaf::io::UTFDataFormatException::~~UTFDataFormatException ( ) throw ()`  
`[inline, virtual]`

**6.862.3 Member Function Documentation**

**6.862.3.1** `virtual UTFDataFormatException* decaf::io::UTFDataFormatException::clone ( )const` `[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns**

A new instance of an Exception object that is a copy of this instance.

Reimplemented from **decaf::io::IOException** (p. 2212).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/UTFDataFormatException.h`

**6.863 decaf::util::UUID Class Reference**

A class that represents an immutable universally unique identifier (**UUID** (p. 4080)).

```
#include <src/main/decaf/util/UUID.h>
```

Inheritance diagram for `decaf::util::UUID`:

## Public Member Functions

- **UUID** (long long mostSigBits, long long leastSigBits)  
*Constructs a new **UUID** (p. 4080) using the specified data.*
- virtual **~UUID** ()
- virtual int **compareTo** (const **UUID** &value) const  
*Compare the given **UUID** (p. 4080) to this one.*
- virtual bool **equals** (const **UUID** &value) const  
*Compares this **UUID** (p. 4080) to the one given, returns true if they are equal.*
- virtual bool **operator==** (const **UUID** &value) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **UUID** &value) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual std::string **toString** () const  
*Returns a String object representing this **UUID** (p. 4080).*
- virtual long long **getLeastSignificantBits** () const
- virtual long long **getMostSignificantBits** () const
- virtual long long **node** () throw ( lang::exceptions::UnsupportedOperationException )  
*The node value associated with this **UUID** (p. 4080).*
- virtual long long **timestamp** () throw ( lang::exceptions::UnsupportedOperationException )  
*The timestamp value associated with this **UUID** (p. 4080).*
- virtual int **clockSequence** () throw ( lang::exceptions::UnsupportedOperationException )  
*The clock sequence value associated with this **UUID** (p. 4080).*
- virtual int **variant** () throw ( lang::exceptions::UnsupportedOperationException )  
*The variant number associated with this **UUID** (p. 4080).*
- virtual int **version** () throw ( lang::exceptions::UnsupportedOperationException )  
*The version number associated with this **UUID** (p. 4080).*

## Static Public Member Functions

- static **UUID randomUUID** ()  
*Static factory to retrieve a type 4 (pseudo randomly generated) **UUID** (p. 4080).*
- static **UUID nameUUIDFromBytes** (const std::vector< char > &name)  
*Static factory to retrieve a type 3 (name based) **UUID** (p. 4080) based on the specified byte array.*
- static **UUID nameUUIDFromBytes** (const char \*name, std::size\_t size)  
*Static factory to retrieve a type 3 (name based) **UUID** (p. 4080) based on the specified byte array.*
- static **UUID fromString** (const std::string &name) throw ( lang::exceptions::IllegalArgumentException )  
*Creates a **UUID** (p. 4080) from the string standard representation as described in the **toString**( ) (p. 4087) method.*

### 6.863.1 Detailed Description

A class that represents an immutable universally unique identifier (**UUID** (p. 4080)). A **UUID** (p. 4080) represents a 128-bit value.

There exist different variants of these global identifiers. The methods of this class are for manipulating the Leach-Salz variant, although the constructors allow the creation of any variant of **UUID** (p. 4080) (described below).

The layout of a variant 2 (Leach-Salz) **UUID** (p. 4080) is as follows: The most significant long consists of the following unsigned fields:

```
0xFFFFFFFF00000000 time_low 0x00000000FFFF0000 time_mid 0x000000000000F000
version 0x00000000000000FF time_hi
```

The least significant long consists of the following unsigned fields:

```
0xC000000000000000 variant 0x3FFF000000000000 clock_seq 0x0000FFFFFFFFFFFF
node
```

The variant field contains a value which identifies the layout of the **UUID** (p. 4080). The bit layout described above is valid only for a **UUID** (p. 4080) with a variant value of 2, which indicates the Leach-Salz variant.

The version field holds a value that describes the type of this **UUID** (p. 4080). There are four different basic types of UUIDs: time-based, DCE security, name-based, and randomly generated UUIDs. These types have a version value of 1, 2, 3 and 4, respectively.

For more information including algorithms used to create UUIDs, see the Internet-Draft UUIDs and GUIDs or the standards body definition at ISO/IEC 11578:1996.

## 6.863.2 Constructor & Destructor Documentation

### 6.863.2.1 decaf::util::UUID::UUID ( long long *mostSigBits*, long long *leastSigBits* )

Constructs a new **UUID** (p. 4080) using the specified data.

*mostSigBits* is used for the most significant 64 bits of the **UUID** (p. 4080) and *leastSigBits* becomes the least significant 64 bits of the **UUID** (p. 4080).

#### Parameters

<i>mostSigBits</i>	
<i>leastSigBits</i>	

### 6.863.2.2 virtual decaf::util::UUID::~~UUID ( ) [virtual]

## 6.863.3 Member Function Documentation

### 6.863.3.1 virtual int decaf::util::UUID::clockSequence ( ) throw ( lang::exceptions::UnsupportedOperationException ) [virtual]

The clock sequence value associated with this **UUID** (p. 4080).

The 14 bit clock sequence value is constructed from the clock sequence field of this **UUID** (p. 4080). The clock sequence field is used to guarantee temporal uniqueness in a time-based **UUID** (p. 4080).

The *clockSequence* value is only meaningful in a time-based **UUID** (p. 4080), which has version type 1. If this **UUID** (p. 4080) is not a time-based **UUID** (p. 4080) then this method throws *UnsupportedOperationException*.

#### Returns

the *clockSequence* associated with a V1 **UUID** (p. 4080)

#### Exceptions

<i>UnsupportedOperationException</i>	
--------------------------------------	--

### 6.863.3.2 virtual int decaf::util::UUID::compareTo ( const UUID & *value* ) const [virtual]

Compare the given **UUID** (p. 4080) to this one.

#### Parameters

<i>value</i>	- the <b>UUID</b> (p. 4080) to compare to
--------------	---

**6.863.3.3** `virtual bool decaf::util::UUID::equals ( const UUID & value ) const` [virtual]

Compares this **UUID** (p. 4080) to the one given, returns true if they are equal.

#### Parameters

<i>value</i>	- the <b>UUID</b> (p. 4080) to compare to.
--------------	--

#### Returns

true if UUIDs are the same.

**6.863.3.4** `static UUID decaf::util::UUID::fromString ( const std::string & name ) throw ( lang::exceptions::IllegalArgumentException )` [static]

Creates a **UUID** (p. 4080) from the string standard representation as described in the `toString()` (p. 4087) method.

#### Parameters

<i>name</i>	- a string to be used to construct a <b>UUID</b> (p. 4080).
-------------	---

#### Returns

type 3 **UUID** (p. 4080)

**6.863.3.5** `virtual long long decaf::util::UUID::getLeastSignificantBits ( ) const` [virtual]

#### Returns

the most significant 64 bits of this UUID's 128 bit value.

**6.863.3.6** `virtual long long decaf::util::UUID::getMostSignificantBits ( ) const` [virtual]

#### Returns

the most significant 64 bits of this UUID's 128 bit value.

**6.863.3.7** `static UUID decaf::util::UUID::nameUUIDFromBytes ( const std::vector< char > & name )` [static]

Static factory to retrieve a type 3 (name based) **UUID** (p. 4080) based on the specified byte array.

#### Parameters

<i>name</i>	- a byte array to be used to construct a <b>UUID</b> (p. 4080).
-------------	---



**Returns**

type 3 **UUID** (p. 4080)

**6.863.3.8** static **UUID** decaf::util::UUID::nameUUIDFromBytes ( const char \* *name*, std::size\_t *size* ) [static]

Static factory to retrieve a type 3 (name based) **UUID** (p. 4080) based on the specified byte array.

**Parameters**

<i>name</i>	- a byte array to be used to construct a <b>UUID</b> (p. 4080).
<i>size</i>	- the size of the byte array, or number of bytes to use.

**Returns**

type 3 **UUID** (p. 4080)

**6.863.3.9** virtual long long decaf::util::UUID::node ( ) throw ( lang::exceptions::UnsupportedOperationException ) [virtual]

The node value associated with this **UUID** (p. 4080).

The 48 bit node value is constructed from the node field of this **UUID** (p. 4080). This field is intended to hold the IEEE 802 address of the machine that generated this **UUID** (p. 4080) to guarantee spatial uniqueness.

The node value is only meaningful in a time-based **UUID** (p. 4080), which has version type 1. If this **UUID** (p. 4080) is not a time-based **UUID** (p. 4080) then this method throws UnsupportedOperationException.

**Returns**

the node value of this **UUID** (p. 4080)

**Exceptions**

<i>UnsupportedOperationException</i>	
--------------------------------------	--

**6.863.3.10** virtual bool decaf::util::UUID::operator< ( const **UUID** & *value* ) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

**Parameters**

<i>value</i>	- the value to be compared to this one.
--------------	---

**Returns**

true if this object is equal to the one passed.

**6.863.3.11** `virtual bool decaf::util::UUID::operator== ( const UUID & value ) const`  
[virtual]

Compares equality between this object and the one passed.

**Parameters**

<i>value</i>	- the value to be compared to this one.
--------------	---

**Returns**

true if this object is equal to the one passed.

**6.863.3.12** `static UUID decaf::util::UUID::randomUUID ( )` [static]

Static factory to retrieve a type 4 (pseudo randomly generated) **UUID** (p. 4080).

The **UUID** (p.4080) is generated using a cryptographically strong pseudo random number generator.

**Returns**

type 4 **UUID** (p. 4080)

**6.863.3.13** `virtual long long decaf::util::UUID::timestamp ( ) throw ( lang::exceptions::UnsupportedOperationException )` [virtual]

The timestamp value associated with this **UUID** (p. 4080).

The 60 bit timestamp value is constructed from the time\_low, time\_mid, and time\_hi fields of this **UUID** (p. 4080). The resulting timestamp is measured in 100-nanosecond units since midnight, October 15, 1582 UTC.

The timestamp value is only meaningful in a time-based **UUID** (p. 4080), which has version type 1. If this **UUID** (p. 4080) is not a time-based **UUID** (p. 4080) then this method throws `UnsupportedOperationException`.

**Returns**

the timestamp associated with a V1 **UUID** (p. 4080)

**Exceptions**

<i>UnsupportedOperation Exception</i>	
---	--

**6.863.3.14** `virtual std::string decaf::util::UUID::toString ( ) const` [virtual]

Returns a String object representing this **UUID** (p. 4080).

UUID's are formatted as: 00112233-4455-6677-8899-AABBCCDDEEFF whose length is 36.

#### Returns

formatted string for this **UUID** (p. 4080)

**6.863.3.15** `virtual int decaf::util::UUID::variant ( ) throw ( lang::exceptions::UnsupportedOperationException )` [virtual]

The variant number associated with this **UUID** (p. 4080).

The variant number describes the layout of the **UUID** (p. 4080). The variant number has the following meaning:

\* 0 Reserved for NCS backward compatibility \* 2 The Leach-Salz variant (used by this class) \* 6 Reserved, Microsoft Corporation backward compatibility \* 7 Reserved for future definition

#### Returns

the variant associated with a V1 **UUID** (p. 4080)

#### Exceptions

<i>UnsupportedOperation Exception</i>	
---	--

**6.863.3.16** `virtual int decaf::util::UUID::version ( ) throw ( lang::exceptions::UnsupportedOperationException )` [virtual]

The version number associated with this **UUID** (p. 4080).

The version number describes how this **UUID** (p. 4080) was generated. The version number has the following meaning:

\* 1 Time-based **UUID** (p. 4080) \* 2 DCE security **UUID** (p. 4080) \* 3 Name-based **UUID** (p. 4080) \* 4 Randomly generated **UUID** (p. 4080)

#### Returns

the version associated with a V1 **UUID** (p. 4080)

**Exceptions**

<i>UnsupportedOperation Exception</i>
---

The documentation for this class was generated from the following file:

- src/main/decaf/util/UUID.h

**6.864 activemq::wireformat::WireFormat Class Reference**

Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.

```
#include <src/main/activemq/wireformat/WireFormat.h>
```

Inheritance diagram for activemq::wireformat::WireFormat:

**Public Member Functions**

- virtual  $\sim$ **WireFormat** ()
- virtual void **marshal** (const **Pointer**< **commands::Command** > &command, const **activemq::transport::Transport** \*transport, **decaf::io::DataOutputStream** \*out)=0 throw ( **decaf::io::IOException** )  
*Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.*
- virtual **Pointer**< **commands::Command** > **unmarshal** (const **activemq::transport::Transport** \*transport, **decaf::io::DataInputStream** \*in)=0 throw ( **decaf::io::IOException** )  
*Stream based unmarshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.*
- virtual void **setVersion** (int version)=0  
*Set the Version.*
- virtual int **getVersion** () const =0  
*Get the Version.*
- virtual bool **hasNegotiator** () const =0  
*Returns true if this **WireFormat** (p. 4088) has a Negotiator that needs to wrap the Transport that uses it.*
- virtual bool **inReceive** () const =0  
*Indicates if the WireFormat object is in the process of receiving a message.*

- virtual **Pointer**< **transport::Transport** > **createNegotiator** (const **Pointer**< **transport::Transport** > &transport)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException )

*If the Transport Provides a Negotiator this method will create and return a new instance of the Negotiator.*

### 6.864.1 Detailed Description

Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.

#### Version

#### Revision:

1.1

### 6.864.2 Constructor & Destructor Documentation

6.864.2.1 virtual activemq::wireformat::WireFormat::~WireFormat ( ) [inline, virtual]

### 6.864.3 Member Function Documentation

6.864.3.1 virtual **Pointer**<transport::Transport>  
activemq::wireformat::WireFormat::createNegotiator ( const  
**Pointer**< **transport::Transport** > & *transport* ) throw ( **decaf::lang::exceptions::UnsupportedOperationException** ) [pure  
virtual]

If the Transport Provides a Negotiator this method will create and return a new instance of the Negotiator.

#### Parameters

<i>transport</i>	- the Transport to Wrap the Negotiator around.
------------------	--

#### Returns

new instance of a **WireFormatNegotiator** (p. 4129) as a **Pointer**<**Transport**> (p. 3032).

#### Exceptions

<i>UnsupportedOperationException</i>	if the <b>WireFormat</b> (p. 4088) doesn't have a Negotiator.
--------------------------------------	---

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2975), and **activemq::wireformat::stomp::StompWireFormat** (p. 3752).

**6.864.3.2** `virtual int activemq::wireformat::WireFormat::getVersion ( ) const [pure virtual]`

Get the Version.

#### Returns

the version of the wire format

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2977), and **activemq::wireformat::stomp::StompWireFormat** (p. 3752).

**6.864.3.3** `virtual bool activemq::wireformat::WireFormat::hasNegotiator ( ) const [pure virtual]`

Returns true if this **WireFormat** (p. 4088) has a Negotiator that needs to wrap the Transport that uses it.

#### Returns

true if the **WireFormat** (p. 4088) provides a Negotiator.

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2977), and **activemq::wireformat::stomp::StompWireFormat** (p. 3753).

**6.864.3.4** `virtual bool activemq::wireformat::WireFormat::inReceive ( ) const [pure virtual]`

Indicates if the WireFormat object is in the process of receiving a message.

This is useful for monitoring inactivity and the **WireFormat** (p. 4088) is processing a large message which takes longer than some configured timeout to unmarshal, the inactivity monitor can query the **WireFormat** (p. 4088) instance to determine if its busy or not and not mark the connection as inactive if so.

#### Returns

true if the **WireFormat** (p. 4088) object is unmarshaling a message.

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2977), and **activemq::wireformat::stomp::StompWireFormat** (p. 3753).

**6.864.3.5** virtual void activemq::wireformat::WireFormat::marshal (   
const Pointer< commands::Command > & *command*,   
const activemq::transport::Transport \* *transport*,   
decaf::io::DataOutputStream \* *out* ) throw ( decaf::io::IOException )   
[pure virtual]

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

#### Parameters

<i>command</i>	The Command to Marshal
<i>transport</i>	The Transport that called this method.
<i>out</i>	The output stream to write the command to.

#### Exceptions

<i>IOException</i>	
--------------------	--

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2979), and **activemq::wireformat::stomp::StompWireFormat** (p. 3753).

**6.864.3.6** virtual void activemq::wireformat::WireFormat::setVersion ( int *version* ) [pure virtual]

Set the Version.

#### Parameters

<i>version</i>	the version of the wire format
----------------	--------------------------------

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2982), and **activemq::wireformat::stomp::StompWireFormat** (p. 3753).

**6.864.3.7** virtual Pointer<commands::Command>   
activemq::wireformat::WireFormat::unmarshal ( const   
activemq::transport::Transport \* *transport*, decaf::io::DataInputStream   
\* *in* ) throw ( decaf::io::IOException ) [pure virtual]

Stream based unmarshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.

Returns a Pointer to the newly unmarshaled Command.

#### Parameters

<i>transport</i>	- Pointer to the transport that is making this request.
<i>in</i>	- the input stream to read the command from.

**Returns**

the newly marshaled Command, caller owns the pointer

**Exceptions**

<i>IOException</i>
--------------------

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2983), and **activemq::wireformat::stomp::StompWireFormat** (p. 3754).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormat.h**

**6.865 activemq::wireformat::WireFormatFactory Class Reference**

The **WireFormatFactory** (p. 4092) is the interface that all **WireFormatFactory** (p. 4092) classes must extend.

```
#include <src/main/activemq/wireformat/WireFormatFactory.h>
```

Inheritance diagram for activemq::wireformat::WireFormatFactory:

**Public Member Functions**

- virtual **~WireFormatFactory** ()
- virtual **Pointer< WireFormat > createWireFormat** (const **decaf::util::Properties** &properties)=0 throw ( **decaf::lang::exceptions::IllegalStateException** )

*Creates a new **WireFormat** (p. 4088) Object passing it a set of properties from which it can obtain any optional settings.*

**6.865.1 Detailed Description**

The **WireFormatFactory** (p. 4092) is the interface that all **WireFormatFactory** (p. 4092) classes must extend. The Factory creates a **WireFormat** (p. 4088) Object based on the properties that are set in the passed **Properties** object.



## 6.865.2 Constructor & Destructor Documentation

6.865.2.1 virtual activemq::wireformat::WireFormatFactory::~WireFormatFactory ( )  
[inline, virtual]

## 6.865.3 Member Function Documentation

6.865.3.1 virtual Pointer<WireFormat> activemq::wireformat::WireFormatFactory::createWireFormat ( const decaf::util::Properties & *properties* ) throw ( decaf::lang::exceptions::IllegalStateException ) [pure virtual]

Creates a new **WireFormat** (p. 4088) Object passing it a set of properties from which it can obtain any optional settings.

### Parameters

<i>properties</i>	- the Properties for this <b>WireFormat</b> (p. 4088)
-------------------	---

### Returns

Pointer to a new instance of a **WireFormat** (p. 4088) object.

Implemented in **activemq::wireformat::openwire::OpenWireFormatFactory** (p. 2985), and **activemq::wireformat::stomp::StompWireFormatFactory** (p. 3755).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormatFactory.h**

## 6.866 activemq::commands::WireFormatInfo Class Reference

```
#include <src/main/activemq/commands/WireFormatInfo.h>
```

Inheritance diagram for activemq::commands::WireFormatInfo:

### Public Member Functions

- **WireFormatInfo** ()
- virtual ~**WireFormatInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **DataStructure** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*

- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual bool **isMarshalAware** () const  
*Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.*
- virtual **decaf::lang::Pointer**< **commands::Command** > **visit** (**activemq::state::CommandVisitor** \*visitor) throw ( **exceptions::ActiveMQException** )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*
- int **getVersion** () const  
*Get the current Wireformat Version.*
- void **setVersion** (int version)  
*Set the current Wireformat Version.*
- long long **getMaxInactivityDuration** () const  
*Returns the currently configured Max Inactivity duration.*
- void **setMaxInactivityDuration** (long long maxInactivityDuration)  
*Sets the Max inactivity duration value.*
- long long **getMaxInactivityDurationInitialDelay** () const  
*Returns the currently configured Max Inactivity Initial Delay duration.*
- void **setMaxInactivityDurationInitialDelay** (long long maxInactivityDurationInitialDelay)  
*Sets the Max inactivity initial delay duration value.*
- bool **isStackTraceEnabled** () const  
*Checks if the stackTraceEnabled flag is on.*
- void **setStackTraceEnabled** (bool stackTraceEnabled)  
*Sets if the stackTraceEnabled flag is on.*

- **bool isTcpNoDelayEnabled () const**  
*Checks if the tcpNoDelayEnabled flag is on.*
- **void setTcpNoDelayEnabled (bool tcpNoDelayEnabled)**  
*Sets if the tcpNoDelayEnabled flag is on.*
- **bool isCacheEnabled () const**  
*Checks if the cacheEnabled flag is on.*
- **void setCacheEnabled (bool cacheEnabled)**  
*Sets if the cacheEnabled flag is on.*
- **int getCacheSize () const**  
*Gets the Cache Size setting.*
- **void setCacheSize (int value)**  
*Sets the Cache Size setting.*
- **bool isTightEncodingEnabled () const**  
*Checks if the tightEncodingEnabled flag is on.*
- **void setTightEncodingEnabled (bool tightEncodingEnabled)**  
*Sets if the tightEncodingEnabled flag is on.*
- **bool isSizePrefixDisabled () const**  
*Checks if the sizePrefixDisabled flag is on.*
- **void setSizePrefixDisabled (bool sizePrefixDisabled)**  
*Sets if the sizePrefixDisabled flag is on.*
- **const std::vector< unsigned char > & getMagic () const**  
*Get the Magic field.*
- **void setMagic (const std::vector< unsigned char > &magic)**  
*Sets the value of the magic field.*
- **const std::vector< unsigned char > & getMarshaledProperties () const**  
*Get the marshalledProperties field.*
- **void setMarshaledProperties (const std::vector< unsigned char > &marshalled-Properties)**  
*Sets the value of the marshalledProperties field.*
- **virtual const util::PrimitiveMap & getProperties () const**  
*Gets the Properties for this *Command* (p. 1227).*

- virtual **util::PrimitiveMap** & **getProperties** ()  
*Gets the Properties for this **Command** (p. 1227).*
- virtual void **setProperties** (const **util::PrimitiveMap** &map)  
*Sets the Properties for this **Command** (p. 1227).*
- bool **isValid** () const  
*Determines if we think this is a Valid **WireFormatInfo** (p. 4093) command.*
- virtual bool **isWireFormatInfo** () const
- virtual void **beforeMarshal** (**wireformat::WireFormat** \*wireFormat **AMQCPP\_UNUSED**) throw ( **decaf::io::IOException** )  
*Handles the marshaling of the objects properties into the internal byte array before the object is marshalled to the wire.*
- virtual void **afterUnmarshal** (**wireformat::WireFormat** \*wireFormat **AMQCPP\_UNUSED**) throw ( **decaf::io::IOException** )  
*Called after unmarshaling is started to cleanup the object being unmarshaled.*

### Static Public Attributes

- static const unsigned char **ID\_WIREFORMATINFO** = 1

### 6.866.1 Constructor & Destructor Documentation

- 6.866.1.1 **activemq::commands::WireFormatInfo::WireFormatInfo** ( )
- 6.866.1.2 virtual **activemq::commands::WireFormatInfo::~~WireFormatInfo** ( ) [virtual]

### 6.866.2 Member Function Documentation

- 6.866.2.1 virtual void **activemq::commands::WireFormatInfo::afterUnmarshal** (**wireformat::WireFormat** \*wireFormat **AMQCPP\_UNUSED** ) throw ( **decaf::io::IOException** ) [virtual]

Called after unmarshaling is started to cleanup the object being unmarshaled.

### Parameters

<i>wireFormat</i>	- the wireformat object to control unmarshaling
-------------------	---

Reimplemented from **activemq::commands::BaseDataStructure** (p. 839).

**6.866.2.2** `virtual void activemq::commands::WireFormatInfo::beforeMarshal ( wireformat::WireFormat *wireFormat AMQCPP_UNUSED ) throw ( decaf::io::IOException )` [virtual]

Handles the marshaling of the objects properties into the internal byte array before the object is marshalled to the wire.

#### Parameters

<i>wireFormat</i>	- the wire formatting controller
-------------------	----------------------------------

Reimplemented from **activemq::commands::BaseDataStructure** (p. 839).

**6.866.2.3** `virtual DataStructure* activemq::commands::WireFormatInfo::cloneDataStructure ( ) const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1714).

**6.866.2.4** `virtual void activemq::commands::WireFormatInfo::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

**6.866.2.5** `virtual bool activemq::commands::WireFormatInfo::equals ( const DataStructure * value ) const` [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 766).

**6.866.2.6** `int activemq::commands::WireFormatInfo::getCacheSize ( ) const`

Gets the Cache Size setting.

#### Returns

currently set cache size.

**6.866.2.7** `virtual unsigned char activemq::commands::WireFormatInfo::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataSetructure** (p. 1713) type copy.

Implements **activemq::commands::DataSetructure** (p. 1717).

**6.866.2.8** `const std::vector<unsigned char>& activemq::commands::WireFormatInfo::getMagic ( ) const [inline]`

Get the Magic field.

#### Returns

const reference to a `std::vector<char>`

**6.866.2.9** `const std::vector<unsigned char>& activemq::commands::WireFormatInfo::getMarshaledProperties ( ) const [inline]`

Get the marshalledProperties field.

#### Returns

const reference to a `std::vector<char>`

**6.866.2.10** `long long activemq::commands::WireFormatInfo::getMaxInactivityDuration ( ) const`

Returns the currently configured Max Inactivity duration.

#### Returns

the set inactivity duration value.

**6.866.2.11** `long long activemq::commands::WireFormatInfo::getMaxInactivityDurationInitialDelay ( ) const`

Returns the currently configured Max Inactivity Initial Delay duration.

**Returns**

the set inactivity duration initial delay value.

**6.866.2.12** `virtual util::PrimitiveMap& activemq::commands::WireFormatInfo::getProperties ( ) [inline, virtual]`

Gets the Properties for this **Command** (p. 1227).

**Returns**

the Properties object for this **Command** (p. 1227).

**6.866.2.13** `virtual const util::PrimitiveMap& activemq::commands::WireFormatInfo::getProperties ( ) const [inline, virtual]`

Gets the Properties for this **Command** (p. 1227).

**Returns**

the Properties object for this **Command** (p. 1227).

**6.866.2.14** `int activemq::commands::WireFormatInfo::getVersion ( ) const [inline]`

Get the current Wireformat Version.

**Returns**

int that identifies the version

**6.866.2.15** `bool activemq::commands::WireFormatInfo::isCacheEnabled ( ) const`

Checks if the cacheEnabled flag is on.

**Returns**

true if the flag is on.

**6.866.2.16** `virtual bool activemq::commands::WireFormatInfo::isMarshalAware ( ) const`  
`[inline, virtual]`

Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.

#### Returns

boolean indicating desire to be in marshaling stages

Reimplemented from `activemq::commands::BaseDataStructure` (p. 840).

**6.866.2.17** `bool activemq::commands::WireFormatInfo::isSizePrefixDisabled ( ) const`

Checks if the `sizePrefixDisabled` flag is on.

#### Returns

true if the flag is on.

**6.866.2.18** `bool activemq::commands::WireFormatInfo::isStackTraceEnabled ( ) const`

Checks if the `stackTraceEnabled` flag is on.

#### Returns

true if the flag is on.

**6.866.2.19** `bool activemq::commands::WireFormatInfo::isTcpNoDelayEnabled ( ) const`

Checks if the `tcpNoDelayEnabled` flag is on.

#### Returns

true if the flag is on.

**6.866.2.20** `bool activemq::commands::WireFormatInfo::isTightEncodingEnabled ( ) const`

Checks if the `tightEncodingEnabled` flag is on.

#### Returns

true if the flag is on.



**6.866.2.21** `bool activemq::commands::WireFormatInfo::isValid ( ) const`

Determines if we think this is a Valid **WireFormatInfo** (p. 4093) command.

**Returns**

true if its valid.

**6.866.2.22** `virtual bool activemq::commands::WireFormatInfo::isWireFormatInfo ( ) const`  
[inline, virtual]**Returns**

answers true to the isWireFormatInfo query

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

**6.866.2.23** `void activemq::commands::WireFormatInfo::setCacheEnabled ( bool cacheEnabled )`

Sets if the cacheEnabled flag is on.

**Parameters**

<i>cacheEnabled</i>	- true to turn flag is on
---------------------	---------------------------

**6.866.2.24** `void activemq::commands::WireFormatInfo::setCacheSize ( int value )`

Sets the Cache Size setting.

**Parameters**

<i>value</i>	- value to set to the cache size.
--------------	-----------------------------------

**6.866.2.25** `void activemq::commands::WireFormatInfo::setMagic ( const std::vector< unsigned char > & magic )` [inline]

Sets the value of the magic field.

**Parameters**

<i>magic</i>	- const std::vector<char>
--------------	---------------------------

**6.866.2.26** `void activemq::commands::WireFormatInfo::setMarshaledProperties ( const  
std::vector< unsigned char > & marshalledProperties ) [inline]`

Sets the value of the `marshalledProperties` field.

#### Parameters

<i>marshalled- Properties</i>	The Byte Array vector that contains the marshaled form of the <b>Message</b> (p. 2596) properties, this is the data sent over the wire.
-----------------------------------	---

**6.866.2.27** `void activemq::commands::WireFormatInfo::setMaxInactivityDuration ( long long  
maxInactivityDuration )`

Sets the Max inactivity duration value.

#### Parameters

<i>maxInactivi- tyDuration</i>	- max time a client can be inactive.
------------------------------------	--------------------------------------

**6.866.2.28** `void activemq::commands::WireFormatInfo::setMaxInactivityDurationInitalDelay ( long long  
maxInactivityDurationInitalDelay )`

Sets the Max inactivity initial delay duration value.

#### Parameters

<i>maxInactivi- tyDura- tionIni- talDelay</i>	- time before the inactivity delay is checked.
---	--

**6.866.2.29** `virtual void activemq::commands::WireFormatInfo::setProperties ( const  
util::PrimitiveMap & map ) [inline, virtual]`

Sets the Properties for this **Command** (p. 1227).

#### Parameters

<i>map</i>	- PrimitiveMap to copy
------------	------------------------

**6.866.2.30** `void activemq::commands::WireFormatInfo::setSizePrefixDisabled ( bool  
sizePrefixDisabled )`

Sets if the `sizePrefixDisabled` flag is on.

**Parameters**

<i>sizePre-fixDisabled</i>	- true to turn flag is on
----------------------------	---------------------------

**6.866.2.31** void activemq::commands::WireFormatInfo::setStackTraceEnabled ( bool *stackTraceEnabled* )

Sets if the stackTraceEnabled flag is on.

**Parameters**

<i>stack-TraceEnabled</i>	- ture to turn flag is on
---------------------------	---------------------------

**6.866.2.32** void activemq::commands::WireFormatInfo::setTcpNoDelayEnabled ( bool *tcpNoDelayEnabled* )

Sets if the tcpNoDelayEnabled flag is on.

**Parameters**

<i>tcpNoDelayEnabled</i>	- ture to turn flag is on
--------------------------	---------------------------

**6.866.2.33** void activemq::commands::WireFormatInfo::setTightEncodingEnabled ( bool *tightEncodingEnabled* )

Sets if the tightEncodingEnabled flag is on.

**Parameters**

<i>tightEncodingEnabled</i>	- true to turn flag is on
-----------------------------	---------------------------

**6.866.2.34** void activemq::commands::WireFormatInfo::setVersion ( int *version* )  
[inline]

Set the current Wireformat Version.

**Parameters**

<i>version</i>	- int that identifies the version
----------------	-----------------------------------

**6.866.2.35** `virtual std::string activemq::commands::WireFormatInfo::toString ( ) const`  
`[virtual]`

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 770).

**6.866.2.36** `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::commands::WireFormatInfo::visit (`  
`activemq::state::CommandVisitor * visitor ) throw (`  
`exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3379) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1232).

### 6.866.3 Field Documentation

**6.866.3.1** `const unsigned char activemq::commands::WireFormatInfo::ID_-`  
`WIREFORMATINFO = 1 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/WireFormatInfo.h`

## 6.867 activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 4104).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/WireFormatInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller**:

- **WireFormatInfoMarshaller ()**
- virtual **~WireFormatInfoMarshaller ()**
- virtual **commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )**  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.867.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 4104).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.867.2 Constructor & Destructor Documentation

6.867.2.1 `activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::WireFormatInfoMarshaller ( ) [inline]`

6.867.2.2 `virtual activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller ( ) [inline, virtual]`

## 6.867.3 Member Function Documentation

6.867.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.867.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.867.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

6.867

activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller

Class Reference

4111

Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.867.3.4** virtual void activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::looseUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.867.3.5** virtual int activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::tightMarshal1 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```

6.867.3.6 virtual void activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```

6.867.3.7 virtual void activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**WireFormatInfoMarshaller.h**



6.868

activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller

Class Reference

6.868 — activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller <sup>4113</sup>

## Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 4109).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/WireFormatInfoMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller:

## Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.868.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 4109).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.868.2 Constructor & Destructor Documentation

6.868.2.1 **activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::WireFormatInfoMarshaller**  
 ( ) [inline]

6.868.2.2 **virtual activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller**  
 ( ) [inline, virtual]

### 6.868.3 Member Function Documentation

6.868.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::createObject (**  
**) const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.868.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::getDataStructureType**  
**( ) const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.868.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure**  
**\* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw (**  
**decaf::io::IOException )** [virtual]

Write a object instance to data output stream.

#### Parameters

6.868

activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller

Class Reference

4115

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.868.3.4** virtual void activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.868.3.5** virtual int activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::tightMarshal1 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.868.3.6  virtual void activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.868.3.7  virtual void activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

6.869

**activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller**

**Class Reference**

4117

- [src/main/activemq/wireformat/openwire/marshal/v6/WireFormatInfoMarshaller.h](#)

## 6.869 **activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller** **Class Reference**

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 4113).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller**:

### Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.869.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p.4113).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.869.2 Constructor & Destructor Documentation

6.869.2.1 **activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::WireFormatInfoMarshaller**  
 ( ) [inline]

6.869.2.2 **virtual activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller**  
 ( ) [inline, virtual]

### 6.869.3 Member Function Documentation

6.869.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::createObject (**  
**) const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.869.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::getDataStructureType**  
**( ) const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.869.3.3 **virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure**  
**\* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw (**  
**decaf::io::IOException)** [virtual]

Write a object instance to data output stream.

6.869

activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller

Class Reference

4119

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.869.3.4** virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::looseUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.869.3.5** virtual int activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::tightMarshal1 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.869.3.6 virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.869.3.7 virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:



6.870

**activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller**

**Class Reference**

**4121**

- `src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h`

## 6.870 **activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller** **Class Reference**

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 4117).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller`:

### Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.870.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p.4117).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.870.2 Constructor & Destructor Documentation

6.870.2.1 **activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::WireFormatInfoMarshaller**  
 ( ) [inline]

6.870.2.2 **virtual activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller**  
 ( ) [inline, virtual]

### 6.870.3 Member Function Documentation

6.870.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::createObject (**  
**) const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.870.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::getDataStructureType**  
**( ) const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.870.3.3 **virtual void activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure**  
**\* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw (**  
**decaf::io::IOException)** [virtual]

Write a object instance to data output stream.

6.870

**activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller**

**Class Reference**

**4123**

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.870.3.4** virtual void **activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.870.3.5** virtual int **activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::tightMarshal1** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.870.3.6  virtual void activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.870.3.7  virtual void activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

6.871

**activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller**

**Class Reference**

**4125**

- `src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h`

## 6.871 **activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller** **Class Reference**

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 4121).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller`:

### Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.871.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p.4121).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.871.2 Constructor & Destructor Documentation

6.871.2.1 **activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::WireFormatInfoMarshaller**  
 ( ) [inline]

6.871.2.2 **virtual activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller**  
 ( ) [inline, virtual]

### 6.871.3 Member Function Documentation

6.871.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::createObject (**  
**) const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.871.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::getDataStructureType**  
**( ) const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.871.3.3 **virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure**  
**\* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw (**  
**decaf::io::IOException )** [virtual]

Write a object instance to data output stream.

6.871

activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller

Class Reference

4127

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.871.3.4** virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::looseUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.871.3.5** virtual int activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::tightMarshal1 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.871.3.6 virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.871.3.7 virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:



6.872

**activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller**

**Class Reference**

**4129**

- `src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h`

## 6.872 **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller** **Class Reference**

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 4125).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/WireFormatInfoMarshaller
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller`:

### Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.872.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p.4125).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change,  
 please see the Java Classes in the activemq-openwire-generator module

### 6.872.2 Constructor & Destructor Documentation

6.872.2.1 **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::WireFormatInfoMarshaller**  
 ( ) [inline]

6.872.2.2 **virtual activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller**  
 ( ) [inline, virtual]

### 6.872.3 Member Function Documentation

6.872.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::createObject (**  
**) const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.872.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::getDataStructureType**  
**( ) const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.872.3.3 **virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure**  
**\* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw (**  
**decaf::io::IOException)** [virtual]

Write a object instance to data output stream.

6.872

**activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller**

**Class Reference**

**4131**

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1675).

**6.872.3.4** virtual void **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::looseUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1683).

**6.872.3.5** virtual int **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::tightMarshal1** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1690).

```
6.872.3.6  virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1698).

```
6.872.3.7  virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1706).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/WireFormatInfoMarshaller.h`

## 6.873 activemq::wireformat::WireFormatNegotiator Class Reference

Defines a **WireFormatNegotiator** (p. 4129) which allows a **WireFormat** (p. 4088) to.

```
#include <src/main/activemq/wireformat/WireFormatNegotiator.h>
```

Inheritance diagram for `activemq::wireformat::WireFormatNegotiator`:

### Public Member Functions

- **WireFormatNegotiator** (const **Pointer**< **transport::Transport** > &next)  
*Constructor.*
- virtual ~**WireFormatNegotiator** ()

#### 6.873.1 Detailed Description

Defines a **WireFormatNegotiator** (p. 4129) which allows a **WireFormat** (p. 4088) to.

#### 6.873.2 Constructor & Destructor Documentation

**6.873.2.1** `activemq::wireformat::WireFormatNegotiator::WireFormatNegotiator ( const Pointer< transport::Transport > & next )` `[inline]`

Constructor.

#### Parameters

<i>next</i>	- the next Transport in the chain
-------------	-----------------------------------

**6.873.2.2** `virtual activemq::wireformat::WireFormatNegotiator::~~WireFormatNegotiator ( )`  
`[inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/WireFormatNegotiator.h`

## 6.874 activemq::wireformat::WireFormatRegistry Class Reference

Registry of all **WireFormat** (p. 4088) Factories that are available to the client at run-time.

```
#include <src/main/activemq/wireformat/WireFormatRegistry.h>
```

### Public Member Functions

- virtual **~WireFormatRegistry** ()
- **WireFormatFactory** \* **findFactory** (const std::string &name) const throw ( decaf::lang::exceptions::NoSuchElementException )
 

*Gets a Registered **WireFormatFactory** (p. 4092) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.*
- void **registerFactory** (const std::string &name, **WireFormatFactory** \*factory) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException )
 

*Registers a new **WireFormatFactory** (p. 4092) with this Registry.*
- void **unregisterFactory** (const std::string &name)
 

*Unregisters the Factory with the given name and deletes that instance of the Factory.*
- std::vector< std::string > **getWireFormatNames** () const
 

*Retrieves a list of the names of all the Registered WireFormat's in this Registry.*

### Static Public Member Functions

- static **WireFormatRegistry** & **getInstance** ()
 

*Gets the single instance of the **WireFormatRegistry** (p. 4129).*

#### 6.874.1 Detailed Description

Registry of all **WireFormat** (p. 4088) Factories that are available to the client at run-time. New WireFormat's must have a factory registered here before a connection attempt is made.

**Since**

3.0

## 6.874.2 Constructor & Destructor Documentation

6.874.2.1 `virtual activemq::wireformat::WireFormatRegistry::~~WireFormatRegistry ( )`  
`[virtual]`

## 6.874.3 Member Function Documentation

6.874.3.1 `WireFormatFactory* activemq::wireformat::WireFormatRegistry::findFactory`  
`( const std::string & name ) const throw ( de-`  
`caf::lang::exceptions::NoSuchElementException`  
`)`

Gets a Registered **WireFormatFactory** (p. 4092) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.

### Parameters

<i>name</i>	The name of the Factory to find in the Registry.
-------------	--

### Returns

the Factory registered under the given name.

### Exceptions

<i>NoSuchElementException</i>	if no factory is registered with that name.
-------------------------------	---

6.874.3.2 `static WireFormatRegistry& ac-`  
`tivemq::wireformat::WireFormatRegistry::getInstance ( )`  
`[static]`

Gets the single instance of the **WireFormatRegistry** (p. 4129).

### Returns

reference to the single instance of this Registry

6.874.3.3 `std::vector<std::string> activemq::wireformat::WireFormatRegistry::getWireFormatNames`  
`( ) const`

Retrieves a list of the names of all the Registered WireFormat's in this Registry.

### Returns

stl vector of strings with all the **WireFormat** (p. 4088) names registered.

**6.874.3.4** `void activemq::wireformat::WireFormatRegistry::registerFactory  
( const std::string & name, WireFormatFactory * factory )  
throw ( decaf::lang::exceptions::IllegalArgumentException,  
decaf::lang::exceptions::NullPointerException )`

Registers a new **WireFormatFactory** (p. 4092) with this Registry.

If a Factory with the given name is already registered it is overwritten with the new one. Once a factory is added to the Registry its lifetime is controlled by the Registry, it will be deleted once the Registry has been deleted.

#### Parameters

<i>name</i>	The name of the new Factory to register.
<i>factory</i>	The new Factory to add to the Registry.

#### Exceptions

<i>IllegalArgumentException</i>	is name is the empty string.
<i>NullPointerException</i>	if the Factory is Null.

**6.874.3.5** `void activemq::wireformat::WireFormatRegistry::unregisterFactory ( const std::string  
& name )`

Unregisters the Factory with the given name and deletes that instance of the Factory.

#### Parameters

<i>name</i>	Name of the Factory to unregister and destroy
-------------	---

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormatRegistry.h**

## 6.875 activemq::transport::inactivity::WriteChecker Class Reference

Runnable class used by the {.

```
#include <src/main/activemq/transport/inactivity/WriteChecker.h>
```

Inheritance diagram for activemq::transport::inactivity::WriteChecker:

#### Public Member Functions

- **WriteChecker (InactivityMonitor \*parent)**



- virtual `~WriteChecker()`
- virtual void `run()`

*Run method - called by the Thread class in the context of the thread.*

### 6.875.1 Detailed Description

Runnable class used by the {}.

#### See also

**InactivityMonitor** (p. 2065)} to make periodic writes to the underlying **transport** (p. 99) if no other write activity is going on in order to more quickly detect failures of the connection to the broker.

#### Since

3.1.0

### 6.875.2 Constructor & Destructor Documentation

**6.875.2.1** `activemq::transport::inactivity::WriteChecker::WriteChecker ( InactivityMonitor * parent )`

**6.875.2.2** `virtual activemq::transport::inactivity::WriteChecker::~~WriteChecker ( )`  
[virtual]

### 6.875.3 Member Function Documentation

**6.875.3.1** `virtual void activemq::transport::inactivity::WriteChecker::run ( )` [virtual]

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 3419).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/inactivity/WriteChecker.h`

## 6.876 decaf::io::Writer Class Reference

```
#include <src/main/decaf/io/Writer.h>
```

Inheritance diagram for decaf::io::Writer:

## Public Member Functions

- **Writer ()**
- virtual **~Writer ()**
- virtual void **write** (char v) throw ( decaf::io::IOException )  
*Writes an single byte char value.*
- virtual void **write** (const std::vector< char > &buffer) throw ( decaf::io::IOException )  
*Writes an array of Chars.*
- virtual void **write** (const char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )  
*Writes a byte array to the output stream.*
- virtual void **write** (const char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes a byte array to the output stream.*
- virtual void **write** (const std::string &str) throw ( decaf::io::IOException )  
*Writes a string.*
- virtual void **write** (const std::string &str, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes a string.*
- virtual **decaf::lang::Appendable & append** (char value) throw ( decaf::io::IOException )  
*Appends the specified character to this Appendable.*
- virtual **decaf::lang::Appendable & append** (const **decaf::lang::CharSequence** \*csq) throw ( decaf::io::IOException )  
*Appends the specified character sequence to this Appendable.*
- virtual **decaf::lang::Appendable & append** (const **decaf::lang::CharSequence** \*csq, int start, int end) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Appends a subsequence of the specified character sequence to this Appendable.*

## Protected Member Functions

- virtual void **doWriteArrayBounded** (const char \*buffer, int size, int offset, int length)=0 throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Override this method to customize the functionality of the method write( char\* buffer, int size, int offset, int length ).*

- virtual void **doWriteChar** (char v) throw ( decaf::io::IOException )
- virtual void **doWriteVector** (const std::vector< char > &buffer) throw ( decaf::io::IOException )
- virtual void **doWriteArray** (const char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )
- virtual void **doWriteString** (const std::string &str) throw ( decaf::io::IOException )
- virtual void **doWriteStringBounded** (const std::string &str, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual **decaf::lang::Appendable** & **doAppendChar** (char value) throw ( decaf::io::IOException )
- virtual **decaf::lang::Appendable** & **doAppendCharSequence** (const **decaf::lang::CharSequence** \*csq) throw ( decaf::io::IOException )
- virtual **decaf::lang::Appendable** & **doAppendCharSequenceStartEnd** (const **decaf::lang::CharSequence** \*csq, int start, int end) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException )

## 6.876.1 Constructor & Destructor Documentation

6.876.1.1 **decaf::io::Writer::Writer** ( )

6.876.1.2 **virtual decaf::io::Writer::~~Writer** ( ) [virtual]

## 6.876.2 Member Function Documentation

6.876.2.1 **virtual decaf::lang::Appendable& decaf::io::Writer::append** ( char *value* ) throw ( decaf::io::IOException ) [virtual]

Appends the specified character to this Appendable.

### Parameters

<i>value</i>	The character to append.
--------------	--------------------------

### Returns

a Reference to this Appendable

### Exceptions

<i>Exception</i>	if an error occurs.
------------------	---------------------

Implements **decaf::lang::Appendable** (p. 734).

**6.876.2.2** `virtual decaf::lang::Appendable& decaf::io::Writer::append ( const decaf::lang::CharSequence * csq ) throw ( decaf::io::IOException )`  
`[virtual]`

Appends the specified character sequence to this Appendable.

#### Parameters

<i>csq</i>	The character sequence from which a subsequence will be appended. If <i>csq</i> is NULL, then characters will be appended as if <i>csq</i> contained the string "null".
------------	---

#### Returns

a Reference to this Appendable.

#### Exceptions

<i>Exception</i>	if an error occurs.
------------------	---------------------

Implements **decaf::lang::Appendable** (p. 735).

**6.876.2.3** `virtual decaf::lang::Appendable& decaf::io::Writer::append ( const decaf::lang::CharSequence * csq, int start, int end ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException )` `[virtual]`

Appends a subsequence of the specified character sequence to this Appendable.

#### Parameters

<i>csq</i>	- The character sequence from which a subsequence will be appended. If <i>csq</i> is NULL, then characters will be appended as if <i>csq</i> contained the string "null".
<i>start</i>	The index of the first character in the subsequence.
<i>end</i>	The index of the character following the last character in the subsequence.

#### Returns

a Reference to this Appendable

#### Exceptions

<i>Exception</i>	if an error occurs.
<i>IndexOutOfBoundsException</i>	<i>start</i> is greater than <i>end</i> , or <i>end</i> is greater than <i>csq.length()</i>

Implements **decaf::lang::Appendable** (p. 734).

- 6.876.2.4 `virtual decaf::lang::Appendable& decaf::io::Writer::doAppendChar ( char value )  
throw ( decaf::io::IOException )` [protected, virtual]
- 6.876.2.5 `virtual decaf::lang::Appendable& decaf::io::Writer::doAppendCharSequence ( const decaf::lang::CharSequence * csq ) throw ( decaf::io::IOException )`  
[protected, virtual]
- 6.876.2.6 `virtual decaf::lang::Appendable& decaf::io::Writer::doAppendCharSequenceStartEnd ( const decaf::lang::CharSequence * csq, int start, int end ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException )` [protected, virtual]
- 6.876.2.7 `virtual void decaf::io::Writer::doWriteArray ( const char * buffer, int size ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )`  
[protected, virtual]
- 6.876.2.8 `virtual void decaf::io::Writer::doWriteArrayBounded ( const char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`  
[protected, pure virtual]

Override this method to customize the functionality of the method `write( char* buffer, int size, int offset, int length )`.

All subclasses must override this method to provide the basic **Writer** (p. 4133) functionality.

Implemented in **decaf::io::OutputStreamWriter** (p. 3001).

- 6.876.2.9 `virtual void decaf::io::Writer::doWriteChar ( char v ) throw ( decaf::io::IOException )` [protected, virtual]
- 6.876.2.10 `virtual void decaf::io::Writer::doWriteString ( const std::string & str ) throw ( decaf::io::IOException )` [protected, virtual]
- 6.876.2.11 `virtual void decaf::io::Writer::doWriteStringBounded ( const std::string & str, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException )`  
[protected, virtual]
- 6.876.2.12 `virtual void decaf::io::Writer::doWriteVector ( const std::vector< char > & buffer ) throw ( decaf::io::IOException )` [protected, virtual]
- 6.876.2.13 `virtual void decaf::io::Writer::write ( char v ) throw ( decaf::io::IOException )`  
[virtual]

Writes an single byte char value.

**Parameters**

<i>v</i>	The value to be written.
----------	--------------------------

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	thrown if an error occurs.
--	----------------------------

**6.876.2.14** `virtual void decaf::io::Writer::write ( const std::string & str ) throw ( decaf::io::IOException )` [virtual]

Writes a string.

**Parameters**

<i>str</i>	The string to be written.
------------	---------------------------

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	thrown if an error occurs.
--	----------------------------

**6.876.2.15** `virtual void decaf::io::Writer::write ( const std::string & str, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Writes a string.

**Parameters**

<i>str</i>	The string to be written.
<i>offset</i>	The position in the array to start writing from.
<i>length</i>	The number of bytes in the array to write.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	thrown if an error occurs.
<i>IndexOutOfBoundsException</i>	if offset+length is greater than the string length.

**6.876.2.16** `virtual void decaf::io::Writer::write ( const char * buffer, int size ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )` [virtual]

Writes a byte array to the output stream.

**Parameters**

<i>buffer</i>	The byte array to write (cannot be NULL).
<i>size</i>	The size in bytes of the buffer passed.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
<i>NullPointerException</i>	if buffer is NULL.

**6.876.2.17** virtual void decaf::io::Writer::write ( const std::vector< char > & *buffer* ) throw ( decaf::io::IOException ) [virtual]

Writes an array of Chars.

**Parameters**

<i>buffer</i>	The array to be written.
---------------	--------------------------

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	thrown if an error occurs.
--	----------------------------

**6.876.2.18** virtual void decaf::io::Writer::write ( const char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Writes a byte array to the output stream.

**Parameters**

<i>buffer</i>	The byte array to write (cannot be NULL).
<i>size</i>	The size in bytes of the buffer passed.
<i>offset</i>	The position in the array to start writing from.
<i>length</i>	The number of bytes in the array to write.

**Exceptions**

<b><i>IOException</i></b> (p. 2210)	if an I/O error occurs.
<i>NullPointerException</i>	if buffer is NULL.
<i>IndexOutOfBoundsException</i>	if offset + length > size of the buffer.

The documentation for this class was generated from the following file:

- `src/main/decaf/io/Writer.h`

## 6.877 decaf::security::auth::x500::X500Principal Class Reference

```
#include <src/main/decaf/security/auth/x500/X500Principal.h>
```

Inheritance diagram for `decaf::security::auth::x500::X500Principal`:

### Public Member Functions

- virtual `~X500Principal()`
- virtual `std::string getName()` const =0  
*Provides the name of this principal.*
- virtual void `getEncoded` (`std::vector< unsigned char > &output`) const =0
- virtual `int hashCode()` const =0

### 6.877.1 Constructor & Destructor Documentation

6.877.1.1 `virtual decaf::security::auth::x500::X500Principal::~X500Principal ( )` [`inline`, `virtual`]

### 6.877.2 Member Function Documentation

6.877.2.1 `virtual void decaf::security::auth::x500::X500Principal::getEncoded ( std::vector< unsigned char > & output )` const [`pure virtual`]

6.877.2.2 `virtual std::string decaf::security::auth::x500::X500Principal::getName ( )` const [`pure virtual`]

Provides the name of this principal.

### Returns

the name of this principal.

Implements `decaf::security::Principal` (p. 3115).

6.877.2.3 `virtual int decaf::security::auth::x500::X500Principal::hashCode ( )` const [`pure virtual`]

The documentation for this class was generated from the following file:



- src/main/decaf/security/auth/x500/X500Principal.h

## 6.878 decaf::security::cert::X509Certificate Class Reference

Base interface for all identity certificates.

```
#include <src/main/decaf/security/cert/X509Certificate.h>
```

Inheritance diagram for decaf::security::cert::X509Certificate:

### Public Member Functions

- virtual **~X509Certificate** ()
- virtual void **checkValidity** () const =0 throw (CertificateExpiredException, CertificateNotYetValidException)
- virtual void **checkValidity** (const **decaf::util::Date** &date) const =0 throw (CertificateExpiredException, CertificateNotYetValidException)
- virtual int **getBasicConstraints** () const =0
- virtual void **getIssuerUniqueID** (std::vector< bool > &output) const =0
- virtual const X500Principal \* **getIssuerX500Principal** () const =0
- virtual void **getKeyUsage** (std::vector< unsigned char > &output) const =0
- virtual Date **getNotAfter** () const =0
- virtual Date **getNotBefore** () const =0
- virtual std::string **getSigAlgName** () const =0
- virtual std::string **getSigAlgOID** () const =0
- virtual void **getSigAlgParams** (std::vector< unsigned char > &output) const =0
- virtual void **getSignature** (std::vector< unsigned char > &output) const =0
- virtual void **getSubjectUniqueID** (std::vector< bool > &output) const =0
- virtual const X500Principal \* **getSubjectX500Principal** () const =0
- virtual void **getTBSCertificate** (std::vector< unsigned char > &output) const =0 throw ( CertificateEncodingException )
- virtual int **getVersion** () const =0

### 6.878.1 Detailed Description

Base interface for all identity certificates.

## 6.878.2 Constructor & Destructor Documentation

6.878.2.1 `virtual decaf::security::cert::X509Certificate::~X509Certificate ( ) [inline, virtual]`

## 6.878.3 Member Function Documentation

6.878.3.1 `virtual void decaf::security::cert::X509Certificate::checkValidity ( ) const throw (CertificateExpiredException, CertificateNotYetValidException) [pure virtual]`

6.878.3.2 `virtual void decaf::security::cert::X509Certificate::checkValidity ( const decaf::util::Date & date ) const throw (CertificateExpiredException, CertificateNotYetValidException) [pure virtual]`

6.878.3.3 `virtual int decaf::security::cert::X509Certificate::getBasicConstraints ( ) const [pure virtual]`

6.878.3.4 `virtual void decaf::security::cert::X509Certificate::getIssuerUniqueID ( std::vector< bool > & output ) const [pure virtual]`

6.878.3.5 `virtual const X500Principal* decaf::security::cert::X509Certificate::getIssuerX500Principal ( ) const [pure virtual]`

6.878.3.6 `virtual void decaf::security::cert::X509Certificate::getKeyUsage ( std::vector< unsigned char > & output ) const [pure virtual]`

6.878.3.7 `virtual Date decaf::security::cert::X509Certificate::getNotAfter ( ) const [pure virtual]`

6.878.3.8 `virtual Date decaf::security::cert::X509Certificate::getNotBefore ( ) const [pure virtual]`

6.878.3.9 `virtual std::string decaf::security::cert::X509Certificate::getSigAlgName ( ) const [pure virtual]`

6.878.3.10 `virtual std::string decaf::security::cert::X509Certificate::getSigAlgOID ( ) const [pure virtual]`

6.878.3.11 `virtual void decaf::security::cert::X509Certificate::getSigAlgParams ( std::vector< unsigned char > & output ) const [pure virtual]`

6.878.3.12 `virtual void decaf::security::cert::X509Certificate::getSignature ( std::vector< unsigned char > & output ) const [pure virtual]`

6.878.3.13 `virtual void decaf::security::cert::X509Certificate::getSubjectUniqueID ( std::vector< bool > & output ) const [pure virtual]`

6.878.3.14 `virtual const X500Principal* decaf::security::cert::X509Certificate::getSubjectX500Principal ( ) const [pure virtual]`

---

6.878.3.15 `virtual void decaf::security::cert::X509Certificate::getTBSCertificate ( std::vector< unsigned char > & output ) const throw ( CertificateEncodingException ) [pure virtual]`

6.878.3.16 `virtual int decaf::security::cert::X509Certificate::getVersion ( ) const [pure virtual]`

Generated on Tue Mar 8 2011 17:10:05 for activemq-cpp-3.2.5 by Doxygen

- src/main/decaf/security/cert/X509Certificate.h

## 6.879 activemq::commands::XATransactionId Class Reference

```
#include <src/main/activemq/commands/XATransactionId.h>
```

Inheritance diagram for activemq::commands::XATransactionId:

### Public Types

- typedef decaf::lang::PointerComparator< XATransactionId > COMPARTOR

### Public Member Functions

- **XATransactionId** ()
- **XATransactionId** (const **XATransactionId** &other)
- virtual ~**XATransactionId** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **XATransactionId** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.*
- virtual int **getFormatId** () const
- virtual void **setFormatId** (int formatId)
- virtual const std::vector< unsigned char > & **getGlobalTransactionId** () const
- virtual std::vector< unsigned char > & **getGlobalTransactionId** ()
- virtual void **setGlobalTransactionId** (const std::vector< unsigned char > &globalTransactionId)
- virtual const std::vector< unsigned char > & **getBranchQualifier** () const

- virtual std::vector< unsigned char > & **getBranchQualifier** ()
- virtual void **setBranchQualifier** (const std::vector< unsigned char > &**branchQualifier**)
- virtual int **compareTo** (const **XATransactionId** &value) const
- virtual bool **equals** (const **XATransactionId** &value) const
- virtual bool **operator==** (const **XATransactionId** &value) const
- virtual bool **operator<** (const **XATransactionId** &value) const
- **XATransactionId** & **operator=** (const **XATransactionId** &other)

### Static Public Attributes

- static const unsigned char **ID\_XATRANSACTIONID** = 112

### Protected Attributes

- int **formatId**
- std::vector< unsigned char > **globalTransactionId**
- std::vector< unsigned char > **branchQualifier**

## 6.879.1 Member Typedef Documentation

6.879.1.1 **typedef decaf::lang::PointerComparator<XATransactionId>  
activemq::commands::XATransactionId::COMPARATOR**

Reimplemented from **activemq::commands::TransactionId** (p. 3933).

## 6.879.2 Constructor & Destructor Documentation

6.879.2.1 **activemq::commands::XATransactionId::XATransactionId ( )**

6.879.2.2 **activemq::commands::XATransactionId::XATransactionId ( const XATransactionId  
& other )**

6.879.2.3 **virtual activemq::commands::XATransactionId::~~XATransactionId ( )**  
[virtual]

## 6.879.3 Member Function Documentation

6.879.3.1 **virtual XATransactionId\* activemq::commands::XATransactionId::cloneDataStructure  
( ) const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Reimplemented from **activemq::commands::TransactionId** (p. 3933).

**6.879.3.2** `virtual int activemq::commands::XATransactionId::compareTo ( const XATransactionId & value ) const` [virtual]

**6.879.3.3** `virtual void activemq::commands::XATransactionId::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::TransactionId** (p. 3934).

**6.879.3.4** `virtual bool activemq::commands::XATransactionId::equals ( const DataStructure * value ) const` [virtual]

Compares the **DataStructure** (p. 1713) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::TransactionId** (p. 3934).

**6.879.3.5** `virtual bool activemq::commands::XATransactionId::equals ( const XATransactionId & value ) const` [virtual]

**6.879.3.6** `virtual const std::vector<unsigned char>& activemq::commands::XATransactionId::getBranchQualifier ( ) const` [virtual]

**6.879.3.7** `virtual std::vector<unsigned char>& activemq::commands::XATransactionId::getBranchQualifier ( )` [virtual]

**6.879.3.8** `virtual unsigned char activemq::commands::XATransactionId::getDataStructureType ( ) const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns**

new **DataStructure** (p. 1713) type copy.

Reimplemented from **activemq::commands::TransactionId** (p. 3934).

- 6.879.3.9 `virtual int activemq::commands::XATransactionId::getFormatId ( ) const`  
[virtual]
- 6.879.3.10 `virtual const std::vector<unsigned char>&`  
`activemq::commands::XATransactionId::getGlobalTransactionId ( ) const`  
[virtual]
- 6.879.3.11 `virtual std::vector<unsigned char>& ac-`  
`tivemq::commands::XATransactionId::getGlobalTransactionId (`  
`)` [virtual]
- 6.879.3.12 `virtual bool activemq::commands::XATransactionId::operator< ( const`  
`XATransactionId & value ) const` [virtual]
- 6.879.3.13 `XATransactionId& activemq::commands::XATransactionId::operator= ( const`  
`XATransactionId & other )`
- 6.879.3.14 `virtual bool activemq::commands::XATransactionId::operator== ( const`  
`XATransactionId & value ) const` [virtual]
- 6.879.3.15 `virtual void activemq::commands::XATransactionId::setBranchQualifier ( const`  
`std::vector< unsigned char > & branchQualifier )` [virtual]
- 6.879.3.16 `virtual void activemq::commands::XATransactionId::setFormatId ( int formatId )`  
[virtual]
- 6.879.3.17 `virtual void activemq::commands::XATransactionId::setGlobalTransactionId ( const`  
`std::vector< unsigned char > & globalTransactionId )` [virtual]
- 6.879.3.18 `virtual std::string activemq::commands::XATransactionId::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1713) such as its type and value of its elements.

**Returns**

formatted string useful for debugging.

Reimplemented from **activemq::commands::TransactionId** (p. 3935).

6.880

**activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller**

**Class Reference**

**4151**

**6.879.4 Field Documentation**

---

**6.879.4.1** `std::vector<unsigned char> activemq::commands::XATransactionId::branchQualifier`  
[protected]

**6.879.4.2** `int activemq::commands::XATransactionId::formatId` [protected]

**6.879.4.3** `std::vector<unsigned char> activemq::commands::XATransactionId::globalTransactionId`  
[protected]

**6.879.4.4** `const unsigned char activemq::commands::XATransactionId::ID_-`  
`XATRANSACTIONID = 112` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/XATransactionId.h`

## 6.880 **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller** **Class Reference**

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 4147).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/XATransactionIdMarshaller
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller`:

### Public Member Functions

- **XATransactionIdMarshaller ()**
- **virtual ~XATransactionIdMarshaller ()**
- **virtual commands::DataStructure \* createObject () const**  
*Creates a new instance of this marshalable type.*
- **virtual unsigned char getDataStructureType () const**  
*Get the Data Structure Type that identifies this Marshaler.*
- **virtual void tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*
- **virtual int tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.880.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p.4147).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.880.2 Constructor & Destructor Documentation

6.880.2.1 **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::XATransactionIdMarshaller**  
( ) [inline]

6.880.2.2 **virtual activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::~~XATransactionIdMarshaller**  
( ) [inline, virtual]

### 6.880.3 Member Function Documentation

6.880.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::createObject** ( ) **const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).



6.880

activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller

Class Reference

4153

6.880.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.880.3.3 virtual void activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** (p. 3957).

6.880.3.4 virtual void activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3957).

```
6.880.3.5  virtual int activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3958).

```
6.880.3.6  virtual void activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3958).

6.881

activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller

Class Reference

4155

6.880.3.7 virtual void activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** (p. 3959).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/XATransactionIdMarshaller.h

## 6.881 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 4151).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/XATransactionIdMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller**:

#### Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.881.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p.4151).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.881.2 Constructor & Destructor Documentation

6.881.2.1 **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::XATransactionIdMarshaller**  
( ) [inline]

6.881.2.2 **virtual activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::~~XATransactionIdMarshaller**  
( ) [inline, virtual]

### 6.881.3 Member Function Documentation

6.881.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

6.881

**activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller**

**Class Reference**

**4157**

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.881.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.881.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3945).

**6.881.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3945).

```
6.881.3.5  virtual int activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3946).

```
6.881.3.6  virtual void activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

6.882

**activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller**

**Class Reference**

**4159**

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3946).

**6.881.3.7** virtual void **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3947).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/XATransactionIdMarshaller.h

## 6.882 **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 4155).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/XATransactionIdMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller**:

## Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.882.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p.4155).  
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



6.882

activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller

Class Reference

4161

## 6.882.2 Constructor & Destructor Documentation

6.882.2.1 `activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::XATransactionIdMarshaller ( ) [inline]`

6.882.2.2 `virtual activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::~~XATransactionIdMarshaller ( ) [inline, virtual]`

## 6.882.3 Member Function Documentation

6.882.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1661).

6.882.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

6.882.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3953).

**6.882.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3953).

**6.882.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

6.882

**activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller**

**Class Reference**

**4163**

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3954).

**6.882.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3954).

**6.882.3.7** `virtual void activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3955).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/XATransactionIdMarshaller.h`

## 6.883 `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` Class Reference

Marshaling code for Open Wire Format for `XATransactionIdMarshaller` (p. 4160).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/XATransactionId
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller`:

### Public Member Functions

- **`XATransactionIdMarshaller ()`**
- virtual **`~XATransactionIdMarshaller ()`**
- virtual **`commands::DataStructure * createObject () const`**  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **`getDataStructureType () const`**  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **`tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`**  
*Un-marshall an object instance from the data input stream.*
- virtual int **`tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **`tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`**  
*Write a object instance to data output stream.*
- virtual void **`looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`**  
*Un-marshall an object instance from the data input stream.*
- virtual void **`looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`**

6.883

**activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller**

**Class Reference**

**4165**

---

*Write a object instance to data output stream.*

### 6.883.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 4160).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.883.2 Constructor & Destructor Documentation

6.883.2.1 **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::XATransactionIdMarshaller**  
( ) [inline]

6.883.2.2 **virtual activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::~~XATransactionIdMarshaller**  
( ) [inline, virtual]

### 6.883.3 Member Function Documentation

6.883.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::createObject** (  
) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.883.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

6.883.3.3 **virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure**  
\* **dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw (  
**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3941).

**6.883.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3941).

**6.883.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

6.883

**activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller****Class Reference****4167****Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3942).

**6.883.3.6** virtual void **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::tightMarshal2**  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure**  
 \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*,  
**utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** )  
 [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3942).

**6.883.3.7** virtual void **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::tightUnmarshal**  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream**  
 \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3943).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/XATransactionIdMarshaller.h`

## 6.884 **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 4164).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/XATransactionId
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller**:

**Public Member Functions**

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*



6.884

**activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller**

**Class Reference**

**4169**

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.884.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 4164).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.884.2 Constructor & Destructor Documentation

6.884.2.1 **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::XATransactionIdMarshaller**  
( ) [inline]

6.884.2.2 **virtual activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::~~XATransactionIdMarshaller**  
( ) [inline, virtual]

### 6.884.3 Member Function Documentation

6.884.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

6.884.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1668).

**6.884.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3949).

**6.884.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3949).

6.884

activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller

Class Reference

4171

```
6.884.3.5 virtual int activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3950).

```
6.884.3.6 virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3950).

```
6.884.3.7 virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream
* bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3951).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/XATransactionIdMarshaller.h`

## 6.885 `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` Class Reference

Marshaling code for Open Wire Format for `XATransactionIdMarshaller` (p. 4168).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/XATransactionId
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller`:

#### Public Member Functions

- `XATransactionIdMarshaller ()`
- `virtual ~XATransactionIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- `virtual unsigned char getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaller.*

6.885

activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller

Class Reference

4173

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.885.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p.4168).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.885.2 Constructor & Destructor Documentation

6.885.2.1 **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::XATransactionIdMarshaller**  
( ) [inline]

6.885.2.2 **virtual activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::~~XATransactionIdMarshaller**  
( ) [inline, virtual]

### 6.885.3 Member Function Documentation

6.885.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1661).

**6.885.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1668).

**6.885.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3937).

**6.885.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

6.885

activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller

Class Reference

4175

<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3937).

```
6.885.3.5 virtual int activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3938).

```
6.885.3.6 virtual void activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
  * dataStructure, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3938).

**6.885.3.7** `virtual void activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3939).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/XATransactionIdMarshaller.h`

**6.886 decaf::util::logging::XMLFormatter Class Reference**

Format a **LogRecord** (p. 2487) into a standard XML format.

```
#include <src/main/decaf/util/logging/XMLFormatter.h>
```

Inheritance diagram for `decaf::util::logging::XMLFormatter`:



## Public Member Functions

- **XMLFormatter** ()
- virtual **~XMLFormatter** ()
- virtual std::string **format** (const **LogRecord** &record) const  
*Converts a **LogRecord** (p. 2487) into an XML string.*
- virtual std::string **getHead** (const **Handler** \*handler)  
*Returns the header string for a set of log records formatted as XML strings, using the output handler's encoding if it is defined, otherwise using the default platform encoding.*
- virtual std::string **getTail** (const **Handler** \*handler)  
*Returns the tail string for a set of log records formatted as XML strings.*

### 6.886.1 Detailed Description

Format a **LogRecord** (p. 2487) into a standard XML format. TODO - Currently only outputs UTF-8 The **XMLFormatter** (p. 4172) can be used with arbitrary character encodings, but it is recommended that it normally be used with UTF-8. The character encoding can be set on the output **Handler** (p. 2042).

#### Since

1.0

### 6.886.2 Constructor & Destructor Documentation

6.886.2.1 **decaf::util::logging::XMLFormatter::XMLFormatter** ( )

6.886.2.2 virtual **decaf::util::logging::XMLFormatter::~~XMLFormatter** ( ) [virtual]

### 6.886.3 Member Function Documentation

6.886.3.1 virtual std::string **decaf::util::logging::XMLFormatter::format** ( const **LogRecord** &  
*record* ) const [virtual]

Converts a **LogRecord** (p. 2487) into an XML string.

#### Parameters

<i>record</i>	The log record to be formatted.
---------------	---------------------------------

#### Returns

the log record formatted as an XML string.

Implements **decaf::util::logging::Formatter** (p. 2028).

**6.886.3.2** `virtual std::string decaf::util::logging::XMLFormatter::getHead ( const Handler * handler ) [virtual]`

Returns the header string for a set of log records formatted as XML strings, using the output handler's encoding if it is defined, otherwise using the default platform encoding.

#### Parameters

<i>handler</i>	The output handler, may be NULL.
----------------	----------------------------------

#### Returns

the header string for log records formatted as XML strings.

**6.886.3.3** `virtual std::string decaf::util::logging::XMLFormatter::getTail ( const Handler * handler ) [virtual]`

Returns the tail string for a set of log records formatted as XML strings.

#### Parameters

<i>handler</i>	The output handler, may be NULL.
----------------	----------------------------------

#### Returns

the tail string for log records formatted as XML strings.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/XMLFormatter.h`

## 6.887 `z_stream_s` Struct Reference

```
#include <src/main/decaf/internal/util/zip/zlib.h>
```

#### Data Fields

- `Bytef * next_in`
- `uInt avail_in`
- `uLong total_in`
- `Bytef * next_out`
- `uInt avail_out`
- `uLong total_out`
- `char * msg`
- `struct internal_state FAR * state`
- `alloc_func zalloc`

- free\_func zfree
- voidpf opaque
- int data\_type
- uLong Adler
- uLong reserved

### 6.887.1 Field Documentation

- 6.887.1.1 uLong z\_stream\_s::Adler
- 6.887.1.2 uInt z\_stream\_s::avail\_in
- 6.887.1.3 uInt z\_stream\_s::avail\_out
- 6.887.1.4 int z\_stream\_s::data\_type
- 6.887.1.5 char\* z\_stream\_s::msg
- 6.887.1.6 Bytef\* z\_stream\_s::next\_in
- 6.887.1.7 Bytef\* z\_stream\_s::next\_out
- 6.887.1.8 voidpf z\_stream\_s::opaque
- 6.887.1.9 uLong z\_stream\_s::reserved
- 6.887.1.10 struct internal\_state FAR\* z\_stream\_s::state
- 6.887.1.11 uLong z\_stream\_s::total\_in
- 6.887.1.12 uLong z\_stream\_s::total\_out
- 6.887.1.13 alloc\_func z\_stream\_s::zalloc
- 6.887.1.14 free\_func z\_stream\_s::zfree

The documentation for this struct was generated from the following file:

- src/main/decaf/internal/util/zip/zlib.h

## 6.888 decaf::util::zip::ZipException Class Reference

```
#include <src/main/decaf/util/zip/ZipException.h>
```

Inheritance diagram for decaf::util::zip::ZipException:

## Public Member Functions

- **ZipException** () throw ()  
*Default Constructor.*
- **ZipException** (const lang::Exception &ex) throw ()  
*Copy Constructor.*
- **ZipException** (const ZipException &ex) throw ()  
*Copy Constructor.*
- **ZipException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **ZipException** (const std::exception \*cause) throw ()  
*Constructor.*
- **ZipException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **ZipException** \* clone () const  
*Clones this exception.*
- virtual ~**ZipException** () throw ()

## 6.888.1 Constructor & Destructor Documentation

### 6.888.1.1 decaf::util::zip::ZipException::ZipException ( ) throw () [inline]

Default Constructor.

### 6.888.1.2 decaf::util::zip::ZipException::ZipException ( const lang::Exception & ex ) throw () [inline]

Copy Constructor.

## Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

**6.888.1.3** decaf::util::zip::ZipException::ZipException ( const ZipException & *ex* ) throw ()  
[inline]

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

**6.888.1.4** decaf::util::zip::ZipException::ZipException ( const char \* *file*, const int *lineNumber*,  
const std::exception \* *cause*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.888.1.5** decaf::util::zip::ZipException::ZipException ( const std::exception \* *cause* ) throw ()  
[inline]

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.888.1.6** decaf::util::zip::ZipException::ZipException ( const char \* *file*, const int *lineNumber*,  
const char \* *msg*, ... ) throw () [inline]

Constructor.

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.888.1.7** `virtual decaf::util::zip::ZipException::~~ZipException ( ) throw () [inline, virtual]`

## **6.888.2 Member Function Documentation**

**6.888.2.1** `virtual ZipException* decaf::util::zip::ZipException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### **Returns**

a new instance of an Exception that is a copy of this one.

Reimplemented from **decaf::io::IOException** (p. 2212).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/ZipException.h`

## Chapter 7

# File Documentation

### 7.1 src/main/activemq/cmsutil/CachedConsumer.h File Reference

```
#include <cms/MessageConsumer.h>
#include <activemq/util/Config.h>
```

#### Data Structures

- class **activemq::cmsutil::CachedConsumer**  
*A cached message consumer contained within a pooled session.*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

### 7.2 src/main/activemq/cmsutil/CachedProducer.h File Reference

```
#include <cms/MessageProducer.h>
#include <activemq/util/Config.h>
```

#### Data Structures

- class **activemq::cmsutil::CachedProducer**

*A cached message producer contained within a pooled session.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.3 src/main/activemq/cmsutil/CmsAccessor.h File Reference

```
#include <cms/ConnectionFactory.h>
#include <activemq/cmsutil/ResourceLifecycleManager.h>
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

## Data Structures

- class **activemq::cmsutil::CmsAccessor**  
*Base class for **activemq.cmsutil.CmsTemplate** (p. 1201) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 1361) to operate on.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.4 src/main/activemq/cmsutil/CmsDestinationAccessor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/cmsutil/CmsAccessor.h>
#include <activemq/cmsutil/DynamicDestinationResolver.h>
```



## Data Structures

- class **activemq::cmsutil::CmsDestinationAccessor**

*Extends the **CmsAccessor** (p. 1183) to add support for resolving destination names.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::cmsutil**

## 7.5 src/main/activemq/cmsutil/CmsTemplate.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/cmsutil/CmsDestinationAccessor.h>
#include <activemq/cmsutil/SessionCallback.h>
#include <activemq/cmsutil/ProducerCallback.h>
#include <activemq/cmsutil/SessionPool.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <cms/ConnectionFactory.h>
#include <cms/DeliveryMode.h>
#include <string>
```

## Data Structures

- class **activemq::cmsutil::CmsTemplate**

***CmsTemplate** (p. 1201) simplifies performing synchronous CMS operations.*

- class **activemq::cmsutil::CmsTemplate::ProducerExecutor**
- class **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor**
- class **activemq::cmsutil::CmsTemplate::SendExecutor**
- class **activemq::cmsutil::CmsTemplate::ReceiveExecutor**
- class **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::cmsutil**

## 7.6 src/main/activemq/cmsutil/DestinationResolver.h File Reference

```
#include <cms/Session.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::cmsutil::DestinationResolver**  
*Resolves a CMS destination name to a `Destination`.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.7 src/main/activemq/cmsutil/DynamicDestinationResolver.h File Reference

```
#include <activemq/cmsutil/DestinationResolver.h>
#include <cms/Session.h>
#include <decaf/util/StlMap.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::cmsutil::DynamicDestinationResolver**  
*Resolves a CMS destination name to a `Destination`.*
- class **activemq::cmsutil::DynamicDestinationResolver::SessionResolver**  
*Manages maps of names to topics and queues for a single session.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.8 src/main/activemq/cmsutil/MessageCreator.h File Reference

```
#include <cms/Session.h>
#include <cms/Message.h>
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::cmsutil::MessageCreator**  
*Creates the user-defined message to be sent by the **CmsTemplate** (p. 1201).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.9 src/main/activemq/cmsutil/PooledSession.h File Reference

```
#include <cms/Session.h>
#include <decaf/util/StlMap.h>
#include <activemq/cmsutil/CachedProducer.h>
#include <activemq/cmsutil/CachedConsumer.h>
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::cmsutil::PooledSession**  
*A pooled session object that wraps around a delegate session.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::cmsutil**

## 7.10 src/main/activemq/cmsutil/ProducerCallback.h File Reference

```
#include <cms/Session.h>
#include <cms/MessageProducer.h>
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::cmsutil::ProducerCallback**

*Callback for sending a message to a CMS destination.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::cmsutil**

## 7.11 src/main/activemq/cmsutil/ResourceLifecycleManager.h File Reference

```
#include <cms/Connection.h>
#include <cms/Session.h>
#include <cms/Destination.h>
#include <cms/MessageProducer.h>
#include <cms/MessageConsumer.h>
#include <activemq/util/Config.h>
#include <decaf/util/StlList.h>
```

## 7.12 src/main/decaf/internal/util/ResourceLifecycleManager.h File Reference 4189

### Data Structures

- class **activemq::cmsutil::ResourceLifecycleManager**

*Manages the lifecycle of a set of CMS resources.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::cmsutil**

## 7.12 src/main/decaf/internal/util/ResourceLifecycleManager.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/StlSet.h>
#include <decaf/internal/util/Resource.h>
```

### Data Structures

- class **decaf::internal::util::ResourceLifecycleManager**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::util**

## 7.13 src/main/activemq/cmsutil/SessionCallback.h File Reference

```
#include <cms/Session.h>
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::cmsutil::SessionCallback**

*Callback for executing any number of operations on a provided CMS Session.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::cmsutil**

## 7.14 src/main/activemq/cmsutil/SessionPool.h File Reference

```
#include <activemq/cmsutil/PooledSession.h>
#include <decaf/util/concurrent/Mutex.h>
#include <cms/Connection.h>
#include <list>
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::cmsutil::SessionPool**

*A pool of CMS sessions from the same connection and with the same acknowledge mode.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::cmsutil**

## 7.15 src/main/activemq/commands/ActiveMQBlobMessage.h File Reference

```
#include <activemq/util/Config.h>
```

## **7.16 src/main/activemq/commands/ActiveMQBytesMessage.h File Reference**

```
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/Message.h>
#include <string>
#include <memory>
```

### **Data Structures**

- class **activemq::commands::ActiveMQBlobMessage**

### **Namespaces**

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## **7.16 src/main/activemq/commands/ActiveMQBytesMessage.h File Reference**

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <decaf/io/ByteArrayOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <cms/BytesMessage.h>
#include <vector>
#include <string>
#include <memory>
```

### **Data Structures**

- class **activemq::commands::ActiveMQBytesMessage**

### **Namespaces**

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.17 src/main/activemq/commands/ActiveMQDestination.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/ActiveMQProperties.h>
#include <cms/Destination.h>
#include <decaf/lang/Pointer.h>
#include <vector>
#include <string>
#include <map>
```

### Data Structures

- class **activemq::commands::ActiveMQDestination**
- struct **activemq::commands::ActiveMQDestination::DestinationFilter**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.18 src/main/activemq/commands/ActiveMQMapMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <activemq/util/PrimitiveMap.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <cms/MapMessage.h>
#include <vector>
#include <string>
```



```
#include <memory>
```

## Data Structures

- class **activemq::commands::ActiveMQMapMessage**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.19 src/main/activemq/commands/ActiveMQMessage.h File Reference

```
#include <cms/Message.h>
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
```

## Data Structures

- class **activemq::commands::ActiveMQMessage**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.20 src/main/activemq/commands/ActiveMQMessageTemplate.h File Reference

```
#include <cms/DeliveryMode.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Message.h>
#include <activemq/core/ActiveMQAckHandler.h>
```

```
#include <activemq/core/ActiveMQConnection.h>
#include <activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h>
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <activemq/util/CMSExceptionSupport.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <cms/IllegalStateException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWriteableException.h>
```

### Data Structures

- class **activemq::commands::ActiveMQMessageTemplate**< T >

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.21 src/main/activemq/commands/ActiveMQObjectMessage.h File Reference

```
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/ObjectMessage.h>
#include <activemq/util/Config.h>
#include <memory>
```

### Data Structures

- class **activemq::commands::ActiveMQObjectMessage**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.22 src/main/activemq/commands/ActiveMQQueue.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Exception.h>
#include <cms/Queue.h>
#include <vector>
#include <string>
#include <memory>
```

### Data Structures

- class **activemq::commands::ActiveMQQueue**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.23 src/main/activemq/commands/ActiveMQStreamMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessage.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/StreamMessage.h>
#include <cms/MessageNotWriteableException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageEOFException.h>
```

```
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/ByteArrayOutputStream.h>
#include <string>
#include <memory>
```

### Data Structures

- class **activemq::commands::ActiveMQStreamMessage**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.24 src/main/activemq/commands/ActiveMQTempDestination.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <cms/Closeable.h>
#include <vector>
#include <string>
```

### Data Structures

- class **activemq::commands::ActiveMQTempDestination**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**

## **7.25 src/main/activemq/commands/ActiveMQTempQueue.h File Reference 4197**

- namespace **activemq::commands**

## **7.25 src/main/activemq/commands/ActiveMQTempQueue.h File Reference**

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <cms/TemporaryQueue.h>
#include <vector>
#include <string>
#include <memory>
```

### **Data Structures**

- class **activemq::commands::ActiveMQTempQueue**

### **Namespaces**

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## **7.26 src/main/activemq/commands/ActiveMQTempTopic.h File Reference**

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <cms/TemporaryTopic.h>
#include <vector>
#include <string>
#include <memory>
```

### **Data Structures**

- class **activemq::commands::ActiveMQTempTopic**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.27 src/main/activemq/commands/ActiveMQTextMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/TextMessage.h>
#include <vector>
#include <string>
#include <memory>
```

## Data Structures

- class **activemq::commands::ActiveMQTextMessage**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.28 src/main/activemq/commands/ActiveMQTopic.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Exception.h>
#include <cms/Topic.h>
#include <vector>
#include <string>
```

```
#include <memory>
```

## Data Structures

- class **activemq::commands::ActiveMQTopic**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.29 src/main/activemq/commands/BaseCommand.h File Reference

```
#include <activemq/util/Config.h>  
#include <activemq/commands/Command.h>
```

## Data Structures

- class **activemq::commands::BaseCommand**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.30 src/main/activemq/commands/BaseDataStructure.h File Reference

```
#include <activemq/util/Config.h>  
#include <activemq/commands/DataStructure.h>  
#include <string>  
#include <sstream>
```

## Data Structures

- class **activemq::commands::BaseDataStructure**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::commands**

## 7.31 src/main/activemq/commands/BooleanExpression.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::commands::BooleanExpression**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.32 src/main/activemq/commands/BrokerError.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/BaseCommand.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```



## Data Structures

- class **activemq::commands::BrokerError**  
*This class represents an Exception sent from the Broker.*
- struct **activemq::commands::BrokerError::StackTraceElement**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.33 src/main/activemq/commands/BrokerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::BrokerId**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.34 src/main/activemq/commands/BrokerInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/BrokerInfo.h>
```

```
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::BrokerInfo**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.35 src/main/activemq/commands/Command.h File Reference

```
#include <string>
#include <activemq/util/Config.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::commands::Command**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::state**
- namespace **activemq::commands**

## 7.36 src/main/activemq/commands/ConnectionControl.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::ConnectionControl**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.37 src/main/activemq/commands/ConnectionError.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerError.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::ConnectionError**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

### 7.38 src/main/activemq/commands/ConnectionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

#### Data Structures

- class **activemq::commands::ConnectionId**

#### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

### 7.39 src/main/activemq/commands/ConnectionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::ConnectionInfo**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.40 src/main/activemq/commands/ConsumerControl.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::ConsumerControl**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.41 src/main/activemq/commands/ConsumerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/SessionId.h>
#include <activemq/util/Config.h>
```

```
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::ConsumerId**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.42 src/main/activemq/commands/ConsumerInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BooleanExpression.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::ConsumerInfo**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.43 src/main/activemq/commands/ControlCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::ControlCommand**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.44 src/main/activemq/commands/DataArrayResponse.h File Reference

```
#include <activemq/commands/DataStructure.h>
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::DataArrayResponse**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.45 src/main/activemq/commands/DataResponse.h File Reference

```
#include <activemq/commands/DataStructure.h>
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::DataResponse**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.46 src/main/activemq/commands/DataStructure.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/MarshalAware.h>
```

## Data Structures

- class **activemq::commands::DataStructure**



## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.47 src/main/activemq/commands/DestinationInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::DestinationInfo**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.48 src/main/activemq/commands/DiscoveryEvent.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::DiscoveryEvent**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.49 src/main/activemq/commands/ExceptionResponse.h File Reference

```
#include <activemq/commands/BrokerError.h>
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::ExceptionResponse**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.50 src/main/activemq/commands/FlushCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
```

```
#include <vector>
```

## Data Structures

- class **activemq::commands::FlushCommand**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.51 src/main/activemq/commands/IntegerResponse.h File Reference

```
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::IntegerResponse**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.52 src/main/activemq/commands/JournalQueueAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/MessageAck.h>
```

```
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::JournalQueueAck**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.53 src/main/activemq/commands/JournalTopicAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::JournalTopicAck**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.54 src/main/activemq/commands/JournalTrace.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::JournalTrace**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.55 src/main/activemq/commands/JournalTransaction.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::JournalTransaction**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.56 src/main/activemq/commands/KeepAliveInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::KeepAliveInfo**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.57 src/main/activemq/commands/LastPartialCommand.h File Reference

```
#include <activemq/commands/PartialCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::LastPartialCommand**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## **7.58 src/main/activemq/commands/LocalTransactionId.h File Reference**

```
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### **Data Structures**

- class **activemq::commands::LocalTransactionId**

### **Namespaces**

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## **7.59 src/main/activemq/commands/Message.h File Reference**

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
```

```
#include <activemq/core/ActiveMQAckHandler.h>
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveMap.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::Message**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**
- namespace **activemq::commands**

## 7.60 src/main/cms/Message.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/DeliveryMode.h>
#include <cms/CMSException.h>
#include <cms/IllegalStateException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotWriteableException.h>
```

### Data Structures

- class **cms::Message**  
*Root of all messages.*

### Namespaces

- namespace **cms**



*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.61 src/main/activemq/commands/MessageAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::MessageAck**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.62 src/main/activemq/commands/MessageDispatch.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/Message.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
```

```
#include <vector>
```

## Data Structures

- class **activemq::commands::MessageDispatch**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.63 src/main/activemq/commands/MessageDispatchNotification.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::MessageDispatchNotification**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.64 src/main/activemq/commands/MessageId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/ProducerInfo.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::MessageId**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.65 src/main/activemq/commands/MessagePull.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::MessagePull**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.66 src/main/activemq/commands/NetworkBridgeFilter.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::NetworkBridgeFilter**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.67 src/main/activemq/commands/PartialCommand.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::PartialCommand**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.68 src/main/activemq/commands/ProducerAck.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::ProducerAck**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.69 src/main/activemq/commands/ProducerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/SessionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
```

```
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::ProducerId**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.70 src/main/activemq/commands/ProducerInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::ProducerInfo**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.71 src/main/activemq/commands/RemoveInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::RemoveInfo**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.72 src/main/activemq/commands/RemoveSubscriptionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::RemoveSubscriptionInfo**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

### 7.73 src/main/activemq/commands/ReplayCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

#### Data Structures

- class **activemq::commands::ReplayCommand**

#### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

### 7.74 src/main/activemq/commands/Response.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

#### Data Structures

- class **activemq::commands::Response**

#### Namespaces

- namespace **activemq**



*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.75 src/main/activemq/commands/SessionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::SessionId**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.76 src/main/activemq/commands/SessionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/commands/SessionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::SessionInfo**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.77 src/main/activemq/commands/ShutdownInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::ShutdownInfo**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.78 src/main/activemq/commands/SubscriptionInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::SubscriptionInfo**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.79 src/main/activemq/commands/TransactionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::TransactionId**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.80 src/main/activemq/commands/TransactionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
```

```
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::TransactionInfo**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.81 src/main/activemq/commands/WireFormatInfo.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/PrimitiveMap.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <vector>
```

### Data Structures

- class **activemq::commands::WireFormatInfo**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.82 src/main/activemq/commands/XATransactionId.h File Reference

```
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
```

```
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::XATransactionId**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.83 src/main/activemq/core/ActiveMQAckHandler.h File Reference

```
#include <cms/CMSException.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::core::ActiveMQAckHandler**  
*Interface class that is used to give CMS Messages an interface to Ack themselves with.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**
- namespace **activemq::core**

## 7.84 src/main/activemq/core/ActiveMQConnection.h File Reference

```
#include <cms/Connection.h>
#include <activemq/util/Config.h>
#include <activemq/core/ActiveMQConnectionMetaData.h>
```

```
#include <activemq/core/Dispatcher.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/TransportListener.h>
#include <decaf/util/Properties.h>
#include <decaf/util/STLMap.h>
#include <decaf/util/STLSet.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <string>
#include <memory>
```

## Data Structures

- class **activemq::core::ActiveMQConnection**

*Concrete connection used for all connectors to the ActiveMQ broker.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.85 src/main/activemq/core/ActiveMQConnectionFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/ConnectionFactory.h>
#include <cms/Connection.h>
```

## 7.86 src/main/activemq/core/ActiveMQConnectionMetaData.h File Reference 4231

### Data Structures

- class **activemq::core::ActiveMQConnectionFactory**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.86 src/main/activemq/core/ActiveMQConnectionMetaData.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/ConnectionMetaData.h>
```

### Data Structures

- class **activemq::core::ActiveMQConnectionMetaData**

*This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 260) class.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.87 src/main/activemq/core/ActiveMQConstants.h File Reference

```
#include <string>
#include <map>
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::core::ActiveMQConstants**

*Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values.*

- class **activemq::core::ActiveMQConstants::StaticInitializer**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.88 src/main/activemq/core/ActiveMQConsumer.h File Reference

```
#include <cms/MessageConsumer.h>
#include <cms/MessageListener.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/MessageAck.h>
#include <activemq/commands/MessageDispatch.h>
#include <activemq/core/Dispatcher.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <activemq/core/RedeliveryPolicy.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/concurrent/Mutex.h>
#include <memory>
```



## Data Structures

- class **activemq::core::ActiveMQConsumer**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.89 src/main/activemq/core/ActiveMQProducer.h File Reference

```
#include <cms/MessageProducer.h>
#include <cms/Message.h>
#include <cms/Destination.h>
#include <cms/DeliveryMode.h>
#include <activemq/util/Config.h>
#include <activemq/util/MemoryUsage.h>
#include <activemq/commands/ProducerInfo.h>
#include <activemq/commands/ProducerAck.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <memory>
```

## Data Structures

- class **activemq::core::ActiveMQProducer**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.90 src/main/activemq/core/ActiveMQQueueBrowser.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/Queue.h>
#include <cms/QueueBrowser.h>
#include <cms/MessageEnumeration.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <string>
```

### Data Structures

- class **activemq::core::ActiveMQQueueBrowser**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**

## 7.91 src/main/activemq/core/ActiveMQSession.h File Reference

```
#include <cms/Session.h>
#include <cms/ExceptionListener.h>
#include <activemq/util/Config.h>
#include <activemq/util/Usage.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/core/ActiveMQTransactionContext.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ConsumerInfo.h>
```

```

#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/core/Dispatcher.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/Properties.h>
#include <string>
#include <memory>

```

## Data Structures

- class **activemq::core::ActiveMQSession**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.92 src/main/activemq/core/ActiveMQSessionExecutor.h File Reference

```

#include <activemq/util/Config.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageDispatch.h>
#include <activemq/threads/Task.h>
#include <activemq/threads/TaskRunner.h>
#include <decaf/lang/Pointer.h>

```

## Data Structures

- class **activemq::core::ActiveMQSessionExecutor**

*Delegate dispatcher for a single session.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.93 src/main/activemq/core/ActiveMQTransactionContext.h File Reference

```
#include <memory>
#include <cms/Message.h>
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/LocalTransactionId.h>
#include <activemq/core/Synchronization.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/util/StlSet.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/Mutex.h>
```

## Data Structures

- class **activemq::core::ActiveMQTransactionContext**

*Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.94 src/main/activemq/core/DispatchData.h File Reference

```
#include <stdlib.h>
#include <memory>
#include <activemq/util/Config.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/Message.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::core::DispatchData**

*Simple POCO that contains the information necessary to route a message to a specified consumer.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.95 src/main/activemq/core/Dispatcher.h File Reference

```
#include <activemq/commands/MessageDispatch.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::core::Dispatcher**

*Interface for an object responsible for dispatching messages to consumers.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**

## 7.96 src/main/activemq/core/MessageDispatchChannel.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/MessageDispatch.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/StlQueue.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::core::MessageDispatchChannel**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**

## 7.97 src/main/activemq/core/policies/DefaultPrefetchPolicy.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/core/PrefetchPolicy.h>
```

## Data Structures

- class **activemq::core::policies::DefaultPrefetchPolicy**

## 7.98 src/main/activemq/core/policies/DefaultRedeliveryPolicy.h File Reference 4239

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**
- namespace **activemq::core::policies**

## 7.98 src/main/activemq/core/policies/DefaultRedeliveryPolicy.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/core/RedeliveryPolicy.h>
```

### Data Structures

- class **activemq::core::policies::DefaultRedeliveryPolicy**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**
- namespace **activemq::core::policies**

## 7.99 src/main/activemq/core/PrefetchPolicy.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/Properties.h>
```

### Data Structures

- class **activemq::core::PrefetchPolicy**  
*Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.100 src/main/activemq/core/RedeliveryPolicy.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/Properties.h>
```

## Data Structures

- class **activemq::core::RedeliveryPolicy**

*Interface for a **RedeliveryPolicy** (p. 3267) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.101 src/main/activemq/core/Synchronization.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
```

## Data Structures

- class **activemq::core::Synchronization**

*Transacted Object **Synchronization** (p. 3826), used to sync the events of a Transaction with the items in the Transaction.*



## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.102 src/main/activemq/exceptions/ActiveMQException.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/CMSException.h>
#include <decaf/lang/Exception.h>
#include <activemq/exceptions/ExceptionDefines.h>
#include <stdarg.h>
#include <sstream>
```

## Data Structures

- class **activemq::exceptions::ActiveMQException**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::exceptions**

## 7.103 src/main/activemq/exceptions/BrokerException.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/BrokerError.h>
#include <sstream>
```

## Data Structures

- class `activemq::exceptions::BrokerException`

## Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::exceptions`

## 7.104 src/main/activemq/exceptions/ExceptionDefines.h File Reference

### Defines

- `#define AMQ_CATCH_RETHROW(type)`  
*Macro for catching and re-throwing an exception of a given type.*
- `#define AMQ_CATCH_EXCEPTION_CONVERT(sourceType, targetType)`  
*Macro for catching an exception of one type and then re-throwing as another type.*
- `#define AMQ_CATCHALL_THROW(type)`  
*A catch-all that throws a known exception.*
- `#define AMQ_CATCHALL_NOTHROW()`  
*A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.*
- `#define AMQ_CATCH_NOTHROW(type)`  
*Macro for catching and re-throwing an exception of a given type.*

### 7.104.1 Define Documentation

#### 7.104.1.1 `#define AMQ_CATCH_EXCEPTION_CONVERT( sourceType, targetType )`

##### Value:

```
catch( sourceType& ex ){ \
    targetType target( ex ); \
    target.setMark( __FILE__, __LINE__ ); \
    throw target; \
}
```

Macro for catching an exception of one type and then re-throwing as another type.

#### Parameters

<i>sourceType</i>	the type of the exception to be caught.
<i>targetType</i>	the type of the exception to be thrown.

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()`, `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`, and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

##### 7.104.1.2 #define AMQ\_CATCH\_NOTHROW( type )

#### Value:

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
}
```

Macro for catching and re-throwing an exception of a given type.

#### Parameters

<i>type</i>	The type of the exception to throw (e.g. <code>ActiveMQException</code> ).
-------------	--

##### 7.104.1.3 #define AMQ\_CATCH\_RETHROW( type )

#### Value:

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
    throw ex; \
}
```

Macro for catching and re-throwing an exception of a given type.

#### Parameters

<i>type</i>	The type of the exception to throw (e.g. <code>ActiveMQException</code> ).
-------------	--

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()`, `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`, and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

##### 7.104.1.4 #define AMQ\_CATCHALL\_NOTHROW( )

#### Value:

```
catch( ... ){ \
```

```

        activemq::exceptions::ActiveMQException ex( __FILE__, __LINE__, \
            "caught unknown exception, not rethrowing" ); \
    }

```

A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.

#### 7.104.1.5 #define AMQ\_CATCHALL\_THROW( type )

##### Value:

```

catch( ... ){ \
    type ex( __FILE__, __LINE__, \
        "caught unknown exception" ); \
    throw ex; \
}

```

A catch-all that throws a known exception.

##### Parameters

<i>type</i>	the type of exception to be thrown.
-------------	-------------------------------------

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray`, `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`, and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

## 7.105 src/main/decaf/lang/exceptions/ExceptionDefines.h File Reference

### Defines

- #define **DECAF\_CATCH\_RETHROW**(type)  
*Macro for catching and rethrowing an exception of a given type.*
- #define **DECAF\_CATCH\_EXCEPTION\_CONVERT**(sourceType, targetType)  
*Macro for catching an exception of one type and then rethrowing as another type.*
- #define **DECAF\_CATCHALL\_THROW**(type)  
*A catch-all that throws a known exception.*
- #define **DECAF\_CATCHALL\_NOTHROW**()  
*A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.*
- #define **DECAF\_CATCH\_NOTHROW**(type)

*Macro for catching and rethrowing an exception of a given type.*

### 7.105.1 Define Documentation

#### 7.105.1.1 #define DECAF\_CATCH\_EXCEPTION\_CONVERT( *sourceType*, *targetType* )

**Value:**

```
catch( sourceType& ex ){ \
    targetType target( &ex ); \
    target.setMark( __FILE__, __LINE__ ); \
    throw target; \
}
```

Macro for catching an exception of one type and then rethrowing as another type.

#### Parameters

<i>sourceType</i>	the type of the exception to be caught.
<i>targetType</i>	the type of the exception to be thrown.

Referenced by decaf::util::PriorityQueue< E >::add().

#### 7.105.1.2 #define DECAF\_CATCH\_NOTHROW( *type* )

**Value:**

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
}
```

Macro for catching and rethrowing an exception of a given type.

#### Parameters

<i>type</i>	The type of the exception to throw (e.g. Exception ).
-------------	---

#### 7.105.1.3 #define DECAF\_CATCH\_RETHROW( *type* )

**Value:**

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
    throw ex; \
}
```

Macro for catching and rethrowing an exception of a given type.

**Parameters**

<i>type</i>	The type of the exception to throw (e.g. Exception ).
-------------	---

Referenced by decaf::util::PriorityQueue< E >::add().

**7.105.1.4 #define DECAF\_CATCHALL\_NOTHROW( )****Value:**

```
catch( ... ){ \
    lang::Exception ex( __FILE__, __LINE__, \
        "caught unknown exception, not rethrowing" ); \
}
```

A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.

**7.105.1.5 #define DECAF\_CATCHALL\_THROW( type )****Value:**

```
catch( ... ){ \
    type ex( __FILE__, __LINE__, \
        "caught unknown exception" ); \
    throw ex; \
}
```

A catch-all that throws a known exception.

**Parameters**

<i>type</i>	the type of exception to be thrown.
-------------	-------------------------------------

Referenced by decaf::util::PriorityQueue< E >::add().

**7.106 src/main/activemq/io/LoggingInputStream.h File Reference**

```
#include <activemq/util/Config.h>
#include <decaf/io/FilterInputStream.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

**Data Structures**

- class **activemq::io::LoggingInputStream**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::io**

## 7.107 src/main/activemq/io/LoggingOutputStream.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/io/FilterOutputStream.h>
#include <decaf/util/logging/LoggerDefines.h>
```

## Data Structures

- class **activemq::io::LoggingOutputStream**  
*OutputStream filter that just logs the data being written.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::io**

## 7.108 src/main/activemq/library/ActiveMQCPP.h File Reference

```
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::library::ActiveMQCPP**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::library**

## 7.109 src/main/activemq/state/CommandVisitor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::state::CommandVisitor**

*Interface for an Object that can visit the various Command Objects that are sent from and to this client.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**
- namespace **activemq::state**

## 7.110 src/main/activemq/state/CommandVisitorAdapter.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/state/CommandVisitor.h>
#include <activemq/core/ActiveMQConstants.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ProducerInfo.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/SessionId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/DestinationInfo.h>
#include <activemq/commands/RemoveSubscriptionInfo.h>
```



```
#include <activemq/commands/Message.h>
#include <activemq/commands/MessageAck.h>
#include <activemq/commands/MessagePull.h>
#include <activemq/commands/TransactionInfo.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/commands/ProducerAck.h>
#include <activemq/commands/MessageDispatch.h>
#include <activemq/commands/MessageDispatchNotification.h>
#include <activemq/commands/ControlCommand.h>
#include <activemq/commands/ConnectionError.h>
#include <activemq/commands/ConnectionControl.h>
#include <activemq/commands/ConsumerControl.h>
#include <activemq/commands/ShutdownInfo.h>
#include <activemq/commands/KeepAliveInfo.h>
#include <activemq/commands/FlushCommand.h>
#include <activemq/commands/BrokerError.h>
#include <activemq/commands/BrokerInfo.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/commands/ReplayCommand.h>
#include <activemq/commands/Response.h>
```

## Data Structures

- class **activemq::state::CommandVisitorAdapter**

*Default Implementation of a **CommandVisitor** (p. 1233) that returns NULL for all calls.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::state**

## 7.111 src/main/activemq/state/ConnectionState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/DestinationInfo.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/commands/LocalTransactionId.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
#include <activemq/state/SessionState.h>
#include <activemq/state/TransactionState.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <memory>
```

### Data Structures

- class **activemq::state::ConnectionState**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::state**

## 7.112 src/main/activemq/state/ConnectionStateTracker.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/state/CommandVisitorAdapter.h>
#include <activemq/state/ConnectionState.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
#include <activemq/state/SessionState.h>
#include <activemq/state/TransactionState.h>
#include <activemq/state/Tracked.h>
#include <activemq/transport/Transport.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::state::ConnectionStateTracker**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::state**

## 7.113 src/main/activemq/state/ConsumerState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ConsumerInfo.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <memory>
```

## Data Structures

- class **activemq::state::ConsumerState**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::state**

### 7.114 src/main/activemq/state/ProducerState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ProducerInfo.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <memory>
```

## Data Structures

- class **activemq::state::ProducerState**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::state**

### 7.115 src/main/activemq/state/SessionState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
```

```
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <string>
#include <memory>
```

### Data Structures

- class **activemq::state::SessionState**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::state**

## 7.116 src/main/activemq/state/Tracked.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Response.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::state::Tracked**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::state**

## 7.117 src/main/activemq/state/TransactionState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
```

```
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlList.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <string>
#include <memory>
```

### Data Structures

- class **activemq::state::TransactionState**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::state**

## 7.118 src/main/activemq/threads/CompositeTask.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/Task.h>
```

### Data Structures

- class **activemq::threads::CompositeTask**

*Represents a single task that can be part of a set of Tasks that are contained in a **CompositeTaskRunner** (p. 1256).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::threads**

## 7.119 src/main/activemq/threads/CompositeTaskRunner.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/TaskRunner.h>
#include <activemq/threads/CompositeTask.h>
#include <decaf/util/StlSet.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::threads::CompositeTaskRunner**  
*A **Task** (p. 3846) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::threads**

## 7.120 src/main/activemq/threads/DedicatedTaskRunner.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/TaskRunner.h>
#include <activemq/threads/Task.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::threads::DedicatedTaskRunner**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::threads**

## 7.121 src/main/activemq/threads/Task.h File Reference

```
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::threads::Task**

*Represents a unit of work that requires one or more iterations to complete.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::threads**

## 7.122 src/main/activemq/threads/TaskRunner.h File Reference

```
#include <activemq/util/Config.h>
```

```
#include <activemq/threads/Task.h>
```

## Data Structures

- class **activemq::threads::TaskRunner**



## 7.123 src/main/activemq/transport/AbstractTransportFactory.h File Reference 4257

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::threads**

## 7.123 src/main/activemq/transport/AbstractTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportFactory.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
```

### Data Structures

- class **activemq::transport::AbstractTransportFactory**  
*Abstract implementation of the **TransportFactory** (p. 4003) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 4003) instances.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**

## 7.124 src/main/activemq/transport/CompositeTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <decaf/net/URI.h>
```

```
#include <decaf/util/List.h>
```

## Data Structures

- class **activemq::transport::CompositeTransport**

*A Composite **Transport** (p. 3996) is a **Transport** (p. 3996) implementation that is composed of several Transports.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**

## 7.125 src/main/activemq/transport/correlator/FutureResponse.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/CountDownLatch.h>
#include <activemq/commands/Response.h>
#include <activemq/exceptions/ActiveMQException.h>
```

## Data Structures

- class **activemq::transport::correlator::FutureResponse**

*A container that holds a response object.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**
- namespace **activemq::transport::correlator**

## 7.126 src/main/activemq/transport/correlator/ResponseCorrelator.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <activemq/transport/correlator/FutureResponse.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <map>
#include <stdio.h>
```

### Data Structures

- class **activemq::transport::correlator::ResponseCorrelator**

*This type of transport filter is responsible for correlating asynchronous responses with requests.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**
- namespace **activemq::transport::correlator**

## 7.127 src/main/activemq/transport/DefaultTransportListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/commands/Command.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::transport::DefaultTransportListener**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**

## 7.128 src/main/activemq/transport/failover/BackupTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/DefaultTransportListener.h>
#include <decaf/net/URI.h>
#include <decaf/lang/Pointer.h>
#include <memory>
```

## Data Structures

- class **activemq::transport::failover::BackupTransport**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

## 7.129 src/main/activemq/transport/failover/BackupTransportPool.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/CompositeTask.h>
```

## 7.130 src/main/activemq/transport/failover/CloseTransportsTask.h File Reference

4261

```
#include <activemq/threads/CompositeTaskRunner.h>
#include <activemq/transport/failover/CloseTransportsTask.h>
#include <activemq/transport/failover/BackupTransport.h>
#include <activemq/transport/failover/URIPool.h>
#include <decaf/lang/Pointer.h>
#include <decaf/io/IOException.h>
#include <decaf/util/StlList.h>
```

### Data Structures

- class **activemq::transport::failover::BackupTransportPool**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

## 7.130 src/main/activemq/transport/failover/CloseTransportsTask.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/CompositeTask.h>
#include <activemq/transport/Transport.h>
#include <decaf/util/StlQueue.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::transport::failover::CloseTransportsTask**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**
- namespace **activemq::transport::failover**

## 7.131 src/main/activemq/transport/failover/FailoverTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/threads/TaskRunner.h>
#include <activemq/threads/CompositeTaskRunner.h>
#include <activemq/state/ConnectionStateTracker.h>
#include <activemq/transport/CompositeTransport.h>
#include <activemq/transport/failover/BackupTransportPool.h>
#include <activemq/transport/failover/CloseTransportsTask.h>
#include <activemq/transport/failover/FailoverTransportListener.h>
#include <activemq/transport/failover/URIPool.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/util/StlList.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/atomic/AtomicReference.h>
#include <decaf/net/URI.h>
#include <decaf/io/IOException.h>
```

### Data Structures

- class **activemq::transport::failover::FailoverTransport**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

## 7.132 src/main/activemq/transport/failover/FailoverTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/AbstractTransportFactory.h>
#include <activemq/transport/Transport.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
```

### Data Structures

- class **activemq::transport::failover::FailoverTransportFactory**  
*Creates an instance of a **FailoverTransport** (p. 1930).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

## 7.133 src/main/activemq/transport/failover/FailoverTransportListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportListener.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::transport::failover::FailoverTransportListener**  
*Utility class used by the **Transport** (p. 3996) to perform the work of responding to events from the active **Transport** (p. 3996).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**
- namespace **activemq::transport::failover**

## 7.134 src/main/activemq/transport/failover/URIPool.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/net/URI.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
```

## Data Structures

- class **activemq::transport::failover::URIPool**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**
- namespace **activemq::transport::failover**

## 7.135 src/main/activemq/transport/inactivity/InactivityMonitor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/Pointer.h>
```



## **7.136 src/main/activemq/transport/inactivity/ReadChecker.h File Reference 4265**

```
#include <decaf/util/Timer.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
```

### **Data Structures**

- class **activemq::transport::inactivity::InactivityMonitor**

### **Namespaces**

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::inactivity**

## **7.136 src/main/activemq/transport/inactivity/ReadChecker.h File Reference**

```
#include <activemq/util/Config.h>
#include <decaf/util/TimerTask.h>
```

### **Data Structures**

- class **activemq::transport::inactivity::ReadChecker**  
*Runnable class that is used by the {.*

### **Namespaces**

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::inactivity**

## 7.137 src/main/activemq/transport/inactivity/WriteChecker.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/TimerTask.h>
```

### Data Structures

- class **activemq::transport::inactivity::WriteChecker**

*Runnable class used by the {}.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**
- namespace **activemq::transport::inactivity**

## 7.138 src/main/activemq/transport/IOTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Thread.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <memory>
```

## 7.139 src/main/activemq/transport/logging/LoggingTransport.h File Reference 4267

### Data Structures

- class **activemq::transport::IOTransport**

*Implementation of the **Transport** (p. 3996) interface that performs marshaling of commands to IO streams.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**

## 7.139 src/main/activemq/transport/logging/LoggingTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::transport::logging::LoggingTransport**

*A transport filter that logs commands as they are sent/received.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**
- namespace **activemq::transport::logging**

## 7.140 src/main/activemq/transport/mock/InternalCommandListener.h File Reference

```
#include <activemq/util/Config.h>
```

```
#include <activemq/transport/mock/ResponseBuilder.h>
#include <activemq/transport/DefaultTransportListener.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <decaf/util/concurrent/CountDownLatch.h>
```

## Data Structures

- class **activemq::transport::mock::InternalCommandListener**

*Listens for Commands sent from the **MockTransport** (p. 2854).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**
- namespace **activemq::transport::mock**

## 7.141 src/main/activemq/transport/mock/MockTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/transport/DefaultTransportListener.h>
#include <activemq/transport/mock/ResponseBuilder.h>
#include <activemq/transport/mock/InternalCommandListener.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
```

```
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <decaf/util/concurrent/CountDownLatch.h>
#include <cms/Message.h>
#include <map>
#include <set>
```

## Data Structures

- class **activemq::transport::mock::MockTransport**

*The **MockTransport** (p. 2854) defines a base level **Transport** (p. 3996) class that is intended to be used in place of an a regular protocol **Transport** (p. 3996) such as TCP.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::mock**

## 7.142 src/main/activemq/transport/mock/MockTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/AbstractTransportFactory.h>
```

## Data Structures

- class **activemq::transport::mock::MockTransportFactory**

*Manufactures MockTransports, which are objects that read from input streams and write to output streams.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**
- namespace **activemq::transport::mock**

### 7.143 src/main/activemq/transport/mock/ResponseBuilder.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
```

#### Data Structures

- class **activemq::transport::mock::ResponseBuilder**  
*Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::mock**

### 7.144 src/main/activemq/transport/tcp/SslTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/tcp/TcpTransport.h>
```

#### Data Structures

- class **activemq::transport::tcp::SslTransport**  
*Transport (p. 3996) for connecting to a Broker using an SSL Socket.*

## 7.145 src/main/activemq/transport/tcp/SslTransportFactory.h File Reference 4271

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

## 7.145 src/main/activemq/transport/tcp/SslTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/tcp/TcpTransportFactory.h>
```

### Data Structures

- class **activemq::transport::tcp::SslTransportFactory**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

## 7.146 src/main/activemq/transport/tcp/TcpTransport.h File Reference

```
#include <activemq/io/LoggingInputStream.h>
#include <activemq/io/LoggingOutputStream.h>
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <decaf/net/Socket.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Pointer.h>
#include <decaf/io/BufferedInputStream.h>
```

```
#include <decaf/io/BufferedOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <memory>
```

## Data Structures

- class **activemq::transport::tcp::TcpTransport**  
*Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 2213).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

## 7.147 src/main/activemq/transport/tcp/TcpTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/AbstractTransportFactory.h>
#include <activemq/exceptions/ActiveMQException.h>
```

## Data Structures

- class **activemq::transport::tcp::TcpTransportFactory**  
*Factory Responsible for creating the **TcpTransport** (p. 3865).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::tcp**



## 7.148 src/main/activemq/transport/Transport.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/io/Closeable.h>
#include <decaf/net/URI.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <typeinfo>
```

### Data Structures

- class **activemq::transport::Transport**  
*Interface for a transport layer for command objects.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::transport**

## 7.149 src/main/activemq/transport/TransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::transport::TransportFactory**  
*Defines the interface for Factories that create Transports or TransportFilters.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**

## 7.150 src/main/activemq/transport/TransportFilter.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <activemq/commands/Command.h>
#include <activemq/transport/TransportListener.h>
#include <decaf/lang/Pointer.h>
#include <typeinfo>
```

## Data Structures

- class **activemq::transport::TransportFilter**  
*A filter on the transport layer.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**

## 7.151 src/main/activemq/transport/TransportListener.h File Reference

```
#include <activemq/util/Config.h>
```

```
#include <activemq/commands/Command.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::transport::TransportListener**  
*A listener of asynchronous exceptions from a command transport object.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**

## 7.152 src/main/activemq/transport/TransportRegistry.h File Reference

```
#include <activemq/util/Config.h>
#include <string>
#include <vector>
#include <activemq/transport/TransportFactory.h>
#include <decaf/util/StlMap.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

## Data Structures

- class **activemq::transport::TransportRegistry**  
*Registry of all **Transport** (p. 3996) Factories that are available to the client at runtime.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**

## 7.153 src/main/activemq/util/ActiveMQProperties.h File Reference

```
#include <map>
#include <string>
#include <sstream>
#include <activemq/util/Config.h>
#include <cms/CMSProperties.h>
#include <decaf/util/Properties.h>
```

### Data Structures

- class **activemq::util::ActiveMQProperties**  
*Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p.3216) object.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::util**

## 7.154 src/main/activemq/util/CMSExceptionSupport.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/CMSException.h>
#include <cms/CMSSecurityException.h>
#include <cms/MessageEOFException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWriteableException.h>
#include <cms/InvalidClientIdException.h>
#include <cms/InvalidDestinationException.h>
```

```
#include <cms/InvalidSelectorException.h>
#include <cms/IllegalStateException.h>
#include <cms/UnsupportedOperationException.h>
#include <decaf/lang/Exception.h>
#include <string>
```

## Data Structures

- class **activemq::util::CMSExceptionSupport**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::util**

## Defines

- #define **AMQ\_CATCH\_ALL\_THROW\_CMSEXCEPTION()**  
*Macro for catching an exception of one type and then re-throwing as a Basic CMSException, good for cases where the method isn't specific about what CMS Exceptions are thrown, bad if you need to throw an exception of MessageNotReadableException for instance.*

### 7.154.1 Define Documentation

#### 7.154.1.1 #define AMQ\_CATCH\_ALL\_THROW\_CMSEXCEPTION( )

Macro for catching an exception of one type and then re-throwing as a Basic CMSException, good for cases where the method isn't specific about what CMS Exceptions are thrown, bad if you need to throw an exception of MessageNotReadableException for instance.

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::acknowledge()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::clearBody()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::clearProperties()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::equals()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getBooleanProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getByteProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getCMSMessageID()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getDoubleProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`

```

>::getFloatProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::getIntProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::getLongProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::getPropertyNames(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::getShortProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::getStringProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::propertyExists(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::setBooleanProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::setByteProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::setCMSDestination(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::setCMSReplyTo(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::setDoubleProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::setFloatProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::setIntProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::setLongProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::setShortProperty(), and activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::setStringProperty().

```

## 7.155 src/main/activemq/util/CompositeData.h File Reference

```

#include <activemq/util/Config.h>
#include <decaf/util/Properties.h>
#include <decaf/util/StlList.h>
#include <decaf/net/URI.h>
#include <decaf/net/URISyntaxException.h>

```

### Data Structures

- class **activemq::util::CompositeData**

*Represents a Composite URI.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::util**

## 7.156 src/main/activemq/util/Config.h File Reference

### Defines

- `#define AMQCPP_API`
- `#define HAVE_UUID_UUID_H`
- `#define HAVE_UUID_T`
- `#define HAVE_PTHREAD_H`

#### 7.156.1 Define Documentation

7.156.1.1 `#define AMQCPP_API`

7.156.1.2 `#define HAVE_PTHREAD_H`

7.156.1.3 `#define HAVE_UUID_T`

7.156.1.4 `#define HAVE_UUID_UUID_H`

## 7.157 src/main/cms/Config.h File Reference

### Defines

- `#define CMS_API`

#### 7.157.1 Define Documentation

7.157.1.1 `#define CMS_API`

## 7.158 src/main/decaf/util/Config.h File Reference

### Defines

- `#define DECAF_API`
- `#define HAVE_UUID_UUID_H`
- `#define HAVE_UUID_T`
- `#define HAVE_PTHREAD_H`
- `#define DECAF_UNUSED`

### 7.158.1 Define Documentation

7.158.1.1 `#define DECAF_API`

7.158.1.2 `#define DECAF_UNUSED`

7.158.1.3 `#define HAVE_PTHREAD_H`

7.158.1.4 `#define HAVE_UUID_T`

7.158.1.5 `#define HAVE_UUID_UUID_H`

## 7.159 src/main/activemq/util/IdGenerator.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/concurrent/Mutex.h>
#include <string>
```

### Data Structures

- class `activemq::util::IdGenerator`
- class `activemq::util::IdGenerator::StaticData`

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::util`

## 7.160 src/main/activemq/util/LongSequenceGenerator.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/concurrent/Mutex.h>
```

### Data Structures

- class `activemq::util::LongSequenceGenerator`  
*This class is used to generate a sequence of long long values that are incremented each time a new value is requested.*



## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::util**

## 7.161 src/main/activemq/util/MarshallingSupport.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <string>
```

## Data Structures

- class **activemq::util::MarshallingSupport**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::util**

## 7.162 src/main/activemq/util/MemoryUsage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/Usage.h>
#include <decaf/util/concurrent/Mutex.h>
```

## Data Structures

- class **activemq::util::MemoryUsage**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::util**

## 7.163 src/main/activemq/util/PrimitiveList.h File Reference

```
#include <string>
#include <vector>
#include <decaf/util/StlList.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <stdio.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <activemq/util/PrimitiveValueConverter.h>
```

## Data Structures

- class **activemq::util::PrimitiveList**

*List of primitives.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::util**

## 7.164 src/main/activemq/util/PrimitiveMap.h File Reference

```
#include <string>
#include <vector>
#include <activemq/util/Config.h>
#include <decaf/util/Config.h>
```

```
#include <decaf/util/StlMap.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <activemq/util/PrimitiveValueConverter.h>
```

## Data Structures

- class **activemq::util::PrimitiveMap**

*Map of named primitives.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::util**

## 7.165 src/main/activemq/util/PrimitiveValueConverter.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <string>
```

## Data Structures

- class **activemq::util::PrimitiveValueConverter**

*Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 3099) from one type to another.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::util**

## 7.166 src/main/activemq/util/PrimitiveValueNode.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/Map.h>
#include <decaf/util/List.h>
```

### Data Structures

- class **activemq::util::PrimitiveValueNode**  
*Class that wraps around a single value of one of the many types.*
- union **activemq::util::PrimitiveValueNode::PrimitiveValue**  
*Define a union type comprised of the various types.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::util**

## 7.167 src/main/activemq/util/URISupport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/CompositeData.h>
#include <decaf/util/Properties.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

### Data Structures

- class **activemq::util::URISupport**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::util**

## 7.168 src/main/activemq/util/Usage.h File Reference

```
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::util::Usage**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::util**

## 7.169 src/main/activemq/wireformat/MarshalAware.h File Reference

```
#include <vector>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::wireformat::MarshalAware**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**

## 7.170 src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/DataStreamMarshaller.h>
#include <activemq/wireformat/openwire/utils/HexTable.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller**

*Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

## 7.171 src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h File Reference

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/util/Config.h>
```

## 7.172

src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h

File Reference

4287

### Data Structures

- class **activemq::wireformat::openwire::marshal::DataStreamMarshaller**

*Base class for all classes that marshal commands for Openwire.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

## 7.172 src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h

### File Reference

```
#include <cms/CMSException.h>
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <activemq/util/PrimitiveMap.h>
#include <activemq/util/PrimitiveList.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/IOException.h>
#include <string>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller**

*This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

## 7.173 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 194).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

  - namespace **activemq::wireformat**
  - namespace **activemq::wireformat::openwire**
  - namespace **activemq::wireformat::openwire::marshal**
  - namespace **activemq::wireformat::openwire::marshal::v1**

## 7.174 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
```



## 7.175 src/main/activemq/wireformat/openwire/marsh- shal/v3/ActiveMQBlobMessageMarshaller.h File

### Reference

4289

---

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 202).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.175 src/main/activemq/wireformat/openwire/marsh/v3/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marsh/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 189).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.176 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 198).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.177 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 206).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.178 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 211).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.179 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 239).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.180 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 256).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.181 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 235).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.182 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
```

## 7.183 src/main/activemq/wireformat/openwire/marsh- shal/v5/ActiveMQBytesMessageMarshaller.h File

### Reference

4295

---

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 243).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.183 src/main/activemq/wireformat/openwire/marsh/v5/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marsh/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller* (p. 247).

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.184 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller* (p. 252).

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*



- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.185 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 328).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.186 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 340).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.187 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 324).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.188 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 332).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.189 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 336).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.190 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

## 7.191 src/main/activemq/wireformat/openwire/marsh- shal/v1/ActiveMQMapMessageMarshaller.h File

### Reference

4301

---

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 344).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.191 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 369).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.192 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 381).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.193 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 364).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.194 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMapMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 373).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.195 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarsh File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```



- class **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 377).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.196 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMapMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 386).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.197 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 397).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.198 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
```

## 7.199 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h File

### Reference

4307

---

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 410).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.199 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 393).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.200 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMessageMarshaller File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 401).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.201 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 405).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.202 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 414).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.203 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMar File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 445).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.204 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 457).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.205 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMar File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 440).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.206 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQObjectMessageMar File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
```



## 7.207 src/main/activemq/wireformat/openwire/marsh shal/v5/ActiveMQObjectMessageMarshaller.h File Reference

4313

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 449).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.207 src/main/activemq/wireformat/openwire/marsh/v5/ActiveMQObjectMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marsh/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 453).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.208 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQObjectMessageMar File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 462).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.209 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 491).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.210 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 504).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.211 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller**

*Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 487).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.212 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller**

*Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 495).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.213 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller**

*Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 499).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.214 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQQueueMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h>
```

## 7.215 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQStreamMessageMarshaller.h File Reference

4319

---

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller**

*Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 508).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.215 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQStreamMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQStreamMessageMarshaller*  
(p. 557).

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.216 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQStreamMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQStreamMessageMarshaller*  
(p. 570).

## Namespaces

- namespace **activemq**



## 7.217 src/main/activemq/wireformat/openwire/marsh- shal/v3/ActiveMQStreamMessageMarshaller.h File

### Reference

4321

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.217 src/main/activemq/wireformat/openwire/marsh/v3/ActiveMQStreamMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marsh/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQStreamMessageMarshaller*  
(p. 553).

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.218 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQStreamMessageMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQStreamMessageMarshaller*  
(p.561).

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.219 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQStreamMessageMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
```

## 7.220 src/main/activemq/wireformat/openwire/marsh- shal/v6/ActiveMQStreamMessageMarshaller.h File

### Reference

4323

```
#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQStreamMessageMarshaller*  
(p. 566).

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.220 src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQStreamMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshall/v6/MessageMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQStreamMessageMarshaller*  
(p. 574).

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.221 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTempDestinationMarshaller*  
(p. 587).

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

**7.222 src/main/activemq/wireformat/openwire/marsh-  
shal/v2/ActiveMQTempDestinationMarshaller.h File  
Reference**

4325

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

**7.222 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h  
File Reference**

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

**Data Structures**

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTempDestinationMarshaller*  
(p. 599).

**Namespaces**

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

**7.223 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h  
File Reference**

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
```

```
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTempDestinationMarshaller*  
(p.582).

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.224 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTempDestinationMarshaller*  
(p. 591).

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.225 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTempDestinationMarshaller*  
(p. 595).

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.226 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTempDestinationMarshaller (p.603).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**



## 7.227 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h File Reference

### 7.227 <sup>4329</sup>src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTempQueueMarshaller (p. 615).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.228 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
```

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTempQueueMarshaller (p. 628).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.229 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestination.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTempQueueMarshaller (p. 611).*

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.230 src/main/activemq/wireformat/openwire/marshall/v4/ActiveMQTempQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshall/v4/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTempQueueMarshaller (p. 619).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.231 `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarsha` File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationM
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller`

*Marshaling code for Open Wire Format for ActiveMQTempQueueMarshaller (p. 624).*

### Namespaces

- namespace `activemq`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v5`

## 7.232 `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempQueueMarsha` File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationM
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
```

### 7.233 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h File

#### Reference

4333

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTempQueueMarshaller (p. 632).*

#### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

### 7.233 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTempTopicMarshaller (p. 649).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.234 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTempTopicMarshaller (p. 658).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.235 **src/main/activemq/wireformat/openwire/mar-**  
**shal/v3/ActiveMQTempTopicMarshaller.h** File

Reference

7.235 **src/main/activemq/wireformat/openwire/marsh**<sup>4335</sup>**al/v3/ActiveMQTempTopicMarshaller.h**

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTempTopicMarshaller (p. 641).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.236 **src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempTopicMarshaller.h**  
**File Reference**

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
```

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTempTopicMarshaller (p. 645).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.237 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationM
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTempTopicMarshaller (p. 654).*



- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.238 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTempTopicMarshaller (p. 662).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.239 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 680).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.240 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
```

## 7.241 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h File

### Reference

4339

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 693).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.241 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 671).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.242 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 676).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.243 src/main/activemq/wireformat/openwire/mar-shal/v5/ActiveMQTextMessageMarshaller.h File

Reference

7.243 <sup>4341</sup> src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTextMessageMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshall/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshall::v5::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 684).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshall**
- namespace **activemq::wireformat::openwire::marshall::v5**

## 7.244 src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshall/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
```

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 688).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.245 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 710).*

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.246 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTopicMarshaller (p. 722).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.247 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 701).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.248 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
```



## 7.249 src/main/activemq/wireformat/openwire/marsh- shal/v5/ActiveMQTopicMarshaller.h File

### Reference

4345

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTopicMarshaller (p. 705).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.249 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTopicMarshaller (p. 714).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.250 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for ActiveMQTopicMarshaller (p. 718).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.251

src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h

File Reference

7.251 <sup>4347</sup> src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**

*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 786).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.252 src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
```

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**

*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 807).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.253 src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**

*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 771).*

7.254

src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h

File Reference

4349

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.254 src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller**

*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 778).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.255 src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller**

*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 793).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.256 src/main/activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
```

## 7.257 src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdMarshaller.h File Reference 4351

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller**

*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 800).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.257 src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 886).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.258 src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 898).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**



## 7.259 src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller**  
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 878).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.260 src/main/activemq/wireformat/openwire/marshal/v4/BrokerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller**  
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 882).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.261 src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller**  
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 890).*

## Namespaces

- namespace **activemq**

## 7.262 src/main/activemq/wireformat/openwire/marshal/v6/BrokerIdMarshaller.h File Reference 4355

---

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.262 src/main/activemq/wireformat/openwire/marshal/v6/BrokerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller**  
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 894).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.263 src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 919).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.264 src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## 7.265

src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfoMarshaller.h

File Reference

4357

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 932).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.265 src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 910).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.266 src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 915).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.267 src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

## 7.268

src/main/activemq/wireformat/openwire/marshal/v6/BrokerInfoMarshaller.h

### File Reference

4359

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 923).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.268 src/main/activemq/wireformat/openwire/marshal/v6/BrokerInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 927).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.269 src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1315).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*



- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.270 src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1328).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.271 src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1307).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.272 src/main/activemq/wireformat/openwire/marshal/v4/ConnectionControlMarshaller

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1311).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.273 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1319).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.274 src/main/activemq/wireformat/openwire/marshal/v6/ConnectionControlMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1324).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.275 src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
```

## 7.276 src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h File

### Reference

4365

---

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1349).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.276 src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1336).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.277 src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1340).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.278 src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for ConnectionErrorMarshaller (p. 1344).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.279 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1353).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.280 src/main/activemq/wireformat/openwire/marshal/v6/ConnectionErrorMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```



## 7.281

src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h

File Reference

4369

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1357).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.281 src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1381).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.282 src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1368).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.283 src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

## 7.284

src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h

### File Reference

4371

---

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1373).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.284 src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1377).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.285 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1385).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

7.286

src/main/activemq/wireformat/openwire/marshal/v6/ConnectionIdMarshaller.h

File Reference

4373

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.286 src/main/activemq/wireformat/openwire/marshal/v6/ConnectionIdMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for ConnectionIdMarshaller (p. 1389).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.287 src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1412).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.288 src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1400).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.289 src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1404).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.290 src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1408).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.291 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```



## 7.292 src/main/activemq/wireformat/openwire/marsh/v6/ConnectionInfoMarshaller.h File

### Reference

4377

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1417).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.292 src/main/activemq/wireformat/openwire/marsh/v6/ConnectionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marsh/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1421).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.293 src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller. File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1459).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.294 src/main/activemq/wireformat/openwire/marshal/v2/ConsumerControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for ConsumerControlMarshaller (p. 1447).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.295 src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1451).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.296 src/main/activemq/wireformat/openwire/marshal/v4/ConsumerControlMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for ConsumerControlMarshaller (p. 1455).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.297 src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for ConsumerControlMarshaller (p. 1464).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.298 src/main/activemq/wireformat/openwire/marshal/v6/ConsumerControlMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1468).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.299 src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

### 7.300

**src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h**

#### File Reference

4383

---

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1489).*

#### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

### 7.300 **src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h**

#### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1477).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.301 src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1481).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*



### 7.302

src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h

#### File Reference

4385

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

### 7.302 src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h

#### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for ConsumerIdMarshaller (p. 1485).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

### 7.303 src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h

#### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1493).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.304 src/main/activemq/wireformat/openwire/marshal/v6/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### 7.305

src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h  
File Reference 4387  
Data Structures

---

- class **activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1497).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.305 src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1522).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

### 7.306 src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1510).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

### 7.307 src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
```

### 7.308

#### src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h File Reference 4389

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1514).*

#### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

### 7.308 src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1518).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.309 src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1527).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

### 7.310

src/main/activemq/wireformat/openwire/marshal/v6/ConsumerInfoMarshaller.h  
File Reference 4391

---

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

### 7.310 src/main/activemq/wireformat/openwire/marshal/v6/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for ConsumerInfoMarshaller (p. 1531).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

### 7.311 src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1552).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.312 src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```



- class **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1539).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.313 src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1544).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

### 7.314 src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h

#### File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1548).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

### 7.315 src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h

#### File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

### 7.316 src/main/activemq/wireformat/openwire/marshal/v6/ControlCommandMarshaller.h File

#### Reference

4395

---

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1556).*

#### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

### 7.316 src/main/activemq/wireformat/openwire/marshal/v6/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1561).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.317 src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1587).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.318 src/main/activemq/wireformat/openwire/marshal/v2/DataArrayResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1574).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.319 src/main/activemq/wireformat/openwire/marshal/v3/DataArrayResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1579).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.320 src/main/activemq/wireformat/openwire/marshal/v4/DataArrayResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1583).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.321 src/main/activemq/wireformat/openwire/marshal/v5/DataArrayResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1592).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.322 src/main/activemq/wireformat/openwire/marshal/v6/DataArrayResponseMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1596).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.323 src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
```



### 7.324

#### src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMarshaller.h File Reference 4401

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1656).*

#### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

### 7.324 src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1643).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.325 src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1647).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

### 7.326

src/main/activemq/wireformat/openwire/marshal/v4/DataResponseMarshaller.h  
File Reference 4403

---

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

### 7.326 src/main/activemq/wireformat/openwire/marshal/v4/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1652).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

### 7.327 src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1635).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.328 src/main/activemq/wireformat/openwire/marshal/v6/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1639).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.329 src/main/activemq/wireformat/openwire/marshal/v1/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1797).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

### 7.330 **src/main/activemq/wireformat/openwire/marshal/v2/DestinationInfoMarshaller.h** File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1784).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

### 7.331 **src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller.h** File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
```

### 7.332 src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller.h File

#### Reference

4407

---

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1789).*

#### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

### 7.332 src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1793).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.333 src/main/activemq/wireformat/openwire/marshal/v5/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1806).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*



- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.334 src/main/activemq/wireformat/openwire/marshal/v6/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1801).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.335 src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1831).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.336 src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryEventMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1819).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.337 src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1823).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

### 7.338 src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller.h

#### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1827).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

### 7.339 src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h

#### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

## 7.340 src/main/activemq/wireformat/openwire/marsh/v6/DiscoveryEventMarshaller.h File

### Reference

4413

---

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1835).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.340 src/main/activemq/wireformat/openwire/marsh/v6/DiscoveryEventMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marsh/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1815).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.341 src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1919).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.342 src/main/activemq/wireformat/openwire/marshal/v2/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1902).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.343 src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1906).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.344 src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```



- class **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1915).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.345 src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1910).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.346 src/main/activemq/wireformat/openwire/marshal/v6/ExceptionResponseMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1898).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.347 src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
```

### 7.348 src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h File

#### Reference

4419

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2019).*

#### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

### 7.348 src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2006).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.349 src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2010).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.350 src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2014).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.351 src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2023).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.352 src/main/activemq/wireformat/openwire/marshal/v6/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 2002).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.353 src/main/activemq/wireformat/openwire/marshal/v1/IntegerResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2180).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

### 7.354 src/main/activemq/wireformat/openwire/marshal/v2/IntegerResponseMarshaller.h

#### File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2167).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

### 7.355 src/main/activemq/wireformat/openwire/marshal/v3/IntegerResponseMarshaller.h

#### File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
```



### 7.356 src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h File

#### Reference

4425

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2171).*

#### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

### 7.356 src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2175).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.357 src/main/activemq/wireformat/openwire/marshal/v5/IntegerResponseMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2184).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.358 src/main/activemq/wireformat/openwire/marshal/v6/IntegerResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2162).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.359 src/main/activemq/wireformat/openwire/marshal/v1/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2248).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.360 src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAckMarshaller.

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2232).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.361 src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2240).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.362 src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller. File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2244).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.363 src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller. File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

### 7.364 src/main/activemq/wireformat/openwire/marshal/v6/JournalQueueAckMarshaller.h File

#### Reference

4431

---

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2236).*

#### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

### 7.364 src/main/activemq/wireformat/openwire/marshal/v6/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2228).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.365 src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2277).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*



- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.366 src/main/activemq/wireformat/openwire/marshal/v2/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2261).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.367 src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2265).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.368 src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2273).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.369 src/main/activemq/wireformat/openwire/marshal/v5/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2256).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.370 src/main/activemq/wireformat/openwire/marshal/v6/JournalTopicAckMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2269).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.371 src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

7.372

src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshaller.h

#### File Reference

4437

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2300).*

#### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.372 src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshaller.h

#### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2283).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.373 src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2288).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

7.374

src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h

File Reference

4439

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

**7.374 src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h**

**File Reference**

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2296).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

**7.375 src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMarshaller.h**

**File Reference**

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2304).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.376 src/main/activemq/wireformat/openwire/marshal/v6/JournalTraceMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```



- class **activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2292).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.377 src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2331).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.378 src/main/activemq/wireformat/openwire/marshal/v2/JournalTransactionMarshaller

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2315).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.379 src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarshaller

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

## 7.380 src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarshaller.h File

### Reference

4443

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2319).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.380 src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2327).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.381 src/main/activemq/wireformat/openwire/marshal/v5/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2323).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.382 src/main/activemq/wireformat/openwire/marshal/v6/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2311).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.383 src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2359).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.384 src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

7.385

src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h  
File Reference 4447  
Data Structures

---

- class **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2342).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.385 src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2347).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

### 7.386 src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2351).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

### 7.387 src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```



7.388

**src/main/activemq/wireformat/openwire/marshal/v6/KeepAliveInfoMarshaller.h**  
**File Reference** **4449**

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2355).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.388 src/main/activemq/wireformat/openwire/marshal/v6/KeepAliveInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2338).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.389 src/main/activemq/wireformat/openwire/marshal/v1/LastPartialCommandMarshaller File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2395).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.390 src/main/activemq/wireformat/openwire/marshal/v2/LastPartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2382).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.391 src/main/activemq/wireformat/openwire/marshal/v3/LastPartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2378).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.392 src/main/activemq/wireformat/openwire/marshal/v4/LastPartialCommandMarshaller File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2391).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.393 src/main/activemq/wireformat/openwire/marshal/v5/LastPartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2386).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.394 src/main/activemq/wireformat/openwire/marshal/v6/LastPartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2374).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.395 src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h>
```

### 7.396 src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarshaller.h File

#### Reference

4455

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2446).*

#### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

### 7.396 src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2429).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.397 src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2433).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*



- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.398 src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2442).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.399 src/main/activemq/wireformat/openwire/marshal/v5/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2437).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.400 src/main/activemq/wireformat/openwire/marshal/v6/LocalTransactionIdMarshaller File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2425).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.401 src/main/activemq/wireformat/openwire/marshall/v1/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MarshallerFactory**  
*Used to create marshallers for a specific version of the wire protocol.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.402 src/main/activemq/wireformat/openwire/marshall/v2/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MarshallerFactory**  
*Used to create marshallers for a specific version of the wire protocol.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.403 src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MarshallerFactory**  
*Used to create marshallers for a specific version of the wire protocol.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.404 src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MarshallerFactory**  
*Used to create marshallers for a specific version of the wire protocol.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.405 src/main/activemq/wireformat/openwire/marshall/v5/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MarshallerFactory**  
*Used to create marshallers for a specific version of the wire protocol.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.406 src/main/activemq/wireformat/openwire/marshall/v6/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MarshallerFactory**

*Used to create marshallers for a specific version of the wire protocol.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.407 src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utis/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2665).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

7.408

src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h

File Reference

4463

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.408 src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2653).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.409 src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2657).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.410 src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```



7.411

src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h

File Reference

4465

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2661).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.411 src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2670).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.412 src/main/activemq/wireformat/openwire/marshal/v6/MessageAckMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2648).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.413 src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
```

## 7.414 src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchMarshaller.h File

### Reference

4467

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2707).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.414 src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2690).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.415 src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2695).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.416 src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2703).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.417 src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2699).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.418 src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2712).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.419 src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchNotificationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2737).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.420 src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchNotification File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2725).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**



7.421 src/main/activemq/wireformat/openwire/marsh  
shal/v3/MessageDispatchNotificationMarshaller.h File

Reference

7.421 src/main/activemq/wireformat/openwire/marsh/v3/MessageDispatchNotificationMarshaller.h 4473

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller*  
(p. 2729).

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.422 src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
```

```
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2733).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.423 src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchNotification File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2742).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.424 src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchNotificationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller*  
(p. 2720).

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.425 src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller**

*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2775).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.426 src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
```

## 7.427

### src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h File Reference 4477

```
#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller**

*Marshaling code for Open Wire Format for MessageIdMarshaller (p. 2755).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.427 src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller**

*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2767).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.428 src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller**

*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2759).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

## 7.429

src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarshaller.h File Reference 4479

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.429 src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller**

*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2763).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.430 src/main/activemq/wireformat/openwire/marshal/v6/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
```

```
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utis/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller**

*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2771).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.431 src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utis/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageMarshaller**



*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2798).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.432 src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageMarshaller**  
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2789).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.433 src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageMarshaller**  
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2785).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.434 src/main/activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

## 7.435 src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h File Reference 4483

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageMarshaller**  
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2793).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.435 src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageMarshaller**  
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2780).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.436 src/main/activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MessageMarshaller**  
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2802).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.437 src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
```

7.438

src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h

#### File Reference

4485

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2845).*

#### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.438 src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h

#### File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2828).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.439 src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2837).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.440

src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h

### File Reference

4487

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.440 src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2841).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.441 src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2833).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.442 src/main/activemq/wireformat/openwire/marshal/v6/MessagePullMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```



- class **activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2850).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.443 src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFilterMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2901).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.444 src/main/activemq/wireformat/openwire/marshal/v2/NetworkBridgeFilterMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2881).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.445 src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

## 7.446 src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h File

### Reference

4491

---

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for NetworkBridgeFilterMarshaller (p. 2893).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.446 src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2897).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.447 src/main/activemq/wireformat/openwire/marshal/v5/NetworkBridgeFilterMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2889).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.448 src/main/activemq/wireformat/openwire/marshal/v6/NetworkBridgeFilterMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for NetworkBridgeFilterMarshaller (p. 2885).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.449 src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3028).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.450 src/main/activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3010).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.451 src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3019).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.452 src/main/activemq/wireformat/openwire/marshal/v4/PartialCommandMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3023).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.453 src/main/activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```



## 7.454 src/main/activemq/wireformat/openwire/mar-shal/v6/PartialCommandMarshaller.h File

### Reference

4497

---

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3014).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.454 src/main/activemq/wireformat/openwire/marshal/v6/PartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 3005).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.455 src/main/activemq/wireformat/openwire/marshal/v1/ProducerAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3150).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

7.456

src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h

File Reference

4499

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.456 src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3128).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.457 src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3137).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.458 src/main/activemq/wireformat/openwire/marshal/v4/ProducerAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

7.459

src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h

File Reference

4501

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3133).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.459 src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3141).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.460 src/main/activemq/wireformat/openwire/marshal/v6/ProducerAckMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3145).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.461 src/main/activemq/wireformat/openwire/marshal/v1/ProducerIdMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

7.462

src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarshaller.h

## File Reference

4503

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3181).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.462 src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3161).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.463 src/main/activemq/wireformat/openwire/marshal/v3/ProducerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3169).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*



## 7.464

src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h

### File Reference

4505

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.464 src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3165).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.465 src/main/activemq/wireformat/openwire/marshal/v5/ProducerIdMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p.3173).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.466 src/main/activemq/wireformat/openwire/marshal/v6/ProducerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

7.467

src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshaller.h

File Reference

4507

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3177).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.467 src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3199).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.468 src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3194).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.469 src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
```

## 7.470

src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h

### File Reference

4509

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3207).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.470 src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3190).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.471 src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3203).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

7.472

src/main/activemq/wireformat/openwire/marshal/v6/ProducerInfoMarshaller.h

File Reference

4511

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.472 src/main/activemq/wireformat/openwire/marshal/v6/ProducerInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3211).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.473 src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3300).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.474 src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```



7.475

src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfoMarshaller.h

File Reference

4513

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3288).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.475 src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3296).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.476 src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3309).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.477 src/main/activemq/wireformat/openwire/marshal/v5/RemoveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

7.478

src/main/activemq/wireformat/openwire/marshal/v6/RemoveInfoMarshaller.h

## File Reference

4515

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3305).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.478 src/main/activemq/wireformat/openwire/marshal/v6/RemoveInfoMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3292).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.479 src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3318).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.480 src/main/activemq/wireformat/openwire/marshal/v2/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3326).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.481 src/main/activemq/wireformat/openwire/marshal/v3/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3322).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.482 src/main/activemq/wireformat/openwire/marshal/v4/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3339).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.483 src/main/activemq/wireformat/openwire/marshal/v5/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3335).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.484 src/main/activemq/wireformat/openwire/marshal/v6/RemoveSubscriptionInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3331).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.485 src/main/activemq/wireformat/openwire/marshal/v1/ReplayCommandMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
```



## 7.486 src/main/activemq/wireformat/openwire/marshal/v2/ReplayCommandMarshaller.h File

### Reference

4521

---

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p.3351).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.486 src/main/activemq/wireformat/openwire/marshal/v2/ReplayCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3355).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.487 src/main/activemq/wireformat/openwire/marshal/v3/ReplayCommandMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3360).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.488 src/main/activemq/wireformat/openwire/marshal/v4/ReplayCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3347).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.489 src/main/activemq/wireformat/openwire/marshal/v5/ReplayCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3368).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.490 src/main/activemq/wireformat/openwire/marshal/v6/ReplayCommandMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3364).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.491 src/main/activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3408).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.492 src/main/activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller**  
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3393).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.493 src/main/activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
```

## 7.494 src/main/activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h File Reference 4527

---

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3403).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.494 src/main/activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3388).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.495 src/main/activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3398).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*



## 7.496 src/main/activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h File Reference 4529

---

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.496 src/main/activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller**  
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3413).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.497 src/main/activemq/wireformat/openwire/marshal/v1/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
```

```
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3501).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.498 src/main/activemq/wireformat/openwire/marshal/v2/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3481).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## **7.499 src/main/activemq/wireformat/openwire/marshal/v3/SessionIdMarshaller.h** **File Reference**

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller**  
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3497).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.500 src/main/activemq/wireformat/openwire/marshal/v4/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller**  
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3485).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.501 src/main/activemq/wireformat/openwire/marshal/v5/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

## 7.502 src/main/activemq/wireformat/openwire/marshal/v6/SessionIdMarshaller.h File Reference 4533

---

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller**  
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3493).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.502 src/main/activemq/wireformat/openwire/marshal/v6/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller**  
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3489).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.503 src/main/activemq/wireformat/openwire/marshal/v1/SessionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.3517).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.504

src/main/activemq/wireformat/openwire/marshal/v2/SessionInfoMarshaller.h

File Reference

7.504 <sup>4535</sup> src/main/activemq/wireformat/openwire/marshal/v2/SessionInfoMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3525).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.505 src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
```

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.3521).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.506 src/main/activemq/wireformat/openwire/marshal/v4/SessionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.3530).*



7.507

src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h

File Reference

4537

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.507 src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3513).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.508 src/main/activemq/wireformat/openwire/marshal/v6/SessionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3508).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.509 src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
```

## 7.510

src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMarshaller.h

### File Reference

4539

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3584).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.510 src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3580).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.511 src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3593).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.512

src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h

File Reference

7.512 <sup>4541</sup> src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3597).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.513 src/main/activemq/wireformat/openwire/marshal/v5/ShutdownInfoMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
```

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3589).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.514 src/main/activemq/wireformat/openwire/marshal/v6/ShutdownInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3576).*

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.515 src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3791).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.516 src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3807).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.517 src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
```



## 7.518 src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h File

### Reference

4545

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3786).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.518 src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3799).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.519 src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3795).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.520 src/main/activemq/wireformat/openwire/marshal/v6/SubscriptionInfoMarshaller.h File Reference

### 7.520 <sup>4547</sup>src/main/activemq/wireformat/openwire/marshal/v6/SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3803).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.521 src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
```

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3939).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.522 src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3943).*

## 7.523

src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h

File Reference

4549

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.523 src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3947).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.524 src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3951).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.525 src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
```

7.526

src/main/activemq/wireformat/openwire/marshall/v6/TransactionIdMarshaller.h

File Reference

4551

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3935).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.526 src/main/activemq/wireformat/openwire/marshall/v6/TransactionIdMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshall/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3955).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.527 src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3968).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**



7.528 src/main/activemq/wireformat/openwire/marsh  
shal/v2/TransactionInfoMarshaller.h File

Reference

7.528 src/main/activemq/wireformat/openwire/marsh/v2/TransactionInfoMarshaller.h 4553

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marsh::v2::TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3985).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marsh**
- namespace **activemq::wireformat::openwire::marsh::v2**

7.529 src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
```

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3972).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.530 src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3981).*

## 7.531 src/main/activemq/wireformat/openwire/marshal/v5/TransactionInfoMarshaller.h File

### Reference Namespaces

4555

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.531 src/main/activemq/wireformat/openwire/marshal/v5/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3964).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.532 src/main/activemq/wireformat/openwire/marshal/v6/TransactionInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3977).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.533 src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
```

## 7.534 src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h File

### Reference

4557

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 4121).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.534 src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 4113).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.535 src/main/activemq/wireformat/openwire/marshal/v3/WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p.4125).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.536 src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h File

Reference

7.536 ~~src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h~~<sup>4559</sup>

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshalsec4::WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 4117).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshalsec4**
- namespace **activemq::wireformat::openwire::marshalsec4**

## 7.537 src/main/activemq/wireformat/openwire/marshalsec5/WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshalsec/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
```

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 4104).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.538 src/main/activemq/wireformat/openwire/marshal/v6/WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 4109).*



- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.539 src/main/activemq/wireformat/openwire/marshal/v1/XATransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for XATransactionIdMarshaller (p.4160).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.540 src/main/activemq/wireformat/openwire/marshal/v2/XATransactionIdMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for XATransactionIdMarshaller (p. 4151).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.541 src/main/activemq/wireformat/openwire/marshal/v3/XATransactionIdMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
```

## 7.542 src/main/activemq/wireformat/openwire/marshal/v4/XATransactionIdMarshaller.h File

### Reference

4563

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for XATransactionIdMarshaller (p.4164).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.542 src/main/activemq/wireformat/openwire/marshal/v4/XATransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for XATransactionIdMarshaller (p.4155).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.543 src/main/activemq/wireformat/openwire/marshal/v5/XATransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for XATransactionIdMarshaller (p. 4168).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.544 src/main/activemq/wireformat/openwire/marshal/v6/XATransactionIdMarshaller.h File

Reference

7.544 <sup>4565</sup> src/main/activemq/wireformat/openwire/marshal/v6/XATransactionIdMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshalsec6::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for XATransactionIdMarshaller (p. 4147).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshalsec6**
- namespace **activemq::wireformat::openwire::marshalsec6::v6**

## 7.545 src/main/activemq/wireformat/openwire/OpenWireFormat.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/WireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
#include <decaf/lang/Pointer.h>
```

```
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <memory>
```

## Data Structures

- class **activemq::wireformat::openwire::OpenWireFormat**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

## 7.546 src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormatFactory.h>
#include <activemq/commands/WireFormatInfo.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
```

## Data Structures

- class **activemq::wireformat::openwire::OpenWireFormatFactory**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

## 7.547 src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/WireFormatNegotiator.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/CountDownLatch.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::wireformat::openwire::OpenWireFormatNegotiator**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

## 7.548 src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/mock/ResponseBuilder.h>
#include <decaf/util/StlQueue.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::wireformat::openwire::OpenWireResponseBuilder**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

### 7.549 src/main/activemq/wireformat/openwire/utils/BooleanStream.h File Reference

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::wireformat::openwire::utils::BooleanStream**  
*Manages the writing and reading of boolean data streams to and from a data source such as a DataInputStream or DataOutputStream.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::utils**

### 7.550 src/main/activemq/wireformat/openwire/utils/HexTable.h File Reference

```
#include <vector>
```



7.551

**src/main/activemq/wireformat/openwire/utls/MessagePropertyInterceptor.h File Reference** **4569**

```
#include <string>

#include <activemq/util/Config.h>

#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

## Data Structures

- class **activemq::wireformat::openwire::utls::HexTable**

*The **HexTable** (p. 2048) class maps hexadecimal strings to the value of an index into the table, i.e.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::utls**

## 7.551 src/main/activemq/wireformat/openwire/utls/MessagePropertyInterceptor.h File Reference

```
#include <activemq/util/Config.h>

#include <activemq/commands/Message.h>

#include <activemq/util/PrimitiveMap.h>

#include <activemq/exceptions/ActiveMQException.h>

#include <decaf/lang/exceptions/NullPointerException.h>
```

## Data Structures

- class **activemq::wireformat::openwire::utls::MessagePropertyInterceptor**

*Used the base **ActiveMQMessage** class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the **OpenWire** Message properties.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::utils**

## 7.552 src/main/activemq/wireformat/stomp/StompCommandConstants.h File Reference

```
#include <cms/Destination.h>
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <string>
#include <map>
```

### Data Structures

- class **activemq::wireformat::stomp::StompCommandConstants**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

## 7.553 src/main/activemq/wireformat/stomp/StompFrame.h File Reference

```
#include <string>
#include <string.h>
#include <map>
#include <decaf/util/Properties.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::wireformat::stomp::StompFrame**

*A Stomp-level message frame that encloses all messages to and from the broker.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**

- namespace **activemq::wireformat::stomp**

## 7.554 src/main/activemq/wireformat/stomp/StompHelper.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <activemq/wireformat/stomp/StompFrame.h>
#include <activemq/commands/Message.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::wireformat::stomp::StompHelper**

*Utility Methods used when marshaling to and from StompFrame's.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

## 7.555 src/main/activemq/wireformat/stomp/StompWireFormat.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormat.h>
#include <activemq/wireformat/stomp/StompFrame.h>
#include <activemq/wireformat/stomp/StompHelper.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::wireformat::stomp::StompWireFormat**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

## 7.556 src/main/activemq/wireformat/stomp/StompWireFormatFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormatFactory.h>
#include <activemq/wireformat/stomp/StompWireFormat.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::wireformat::stomp::StompWireFormatFactory**

*Factory used to create the Stomp Wire Format instance.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

## 7.557 src/main/activemq/wireformat/WireFormat.h File Reference

```
#include <activemq/wireformat/WireFormatNegotiator.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/Pointer.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/transport/Transport.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

## Data Structures

- class **activemq::wireformat::WireFormat**

*Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**

## 7.558 src/main/activemq/wireformat/WireFormatFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

### Data Structures

- class **activemq::wireformat::WireFormatFactory**

*The **WireFormatFactory** (p. 4092) is the interface that all **WireFormatFactory** (p. 4092) classes must extend.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**

## 7.559 src/main/activemq/wireformat/WireFormatNegotiator.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::wireformat::WireFormatNegotiator**

*Defines a **WireFormatNegotiator** (p. 4129) which allows a **WireFormat** (p. 4088) to.*

### Namespaces

- namespace **activemq**

## 7.560 src/main/activemq/wireformat/WireFormatRegistry.h File Reference 4575

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**

## 7.560 src/main/activemq/wireformat/WireFormatRegistry.h File Reference

```
#include <activemq/util/Config.h>
#include <string>
#include <vector>
#include <activemq/wireformat/WireFormatFactory.h>
#include <decaf/util/StlMap.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

### Data Structures

- class **activemq::wireformat::WireFormatRegistry**  
*Registry of all **WireFormat** (p. 4088) Factories that are available to the client at run-time.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**

## 7.561 src/main/cms/BytesMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWritableException.h>
```

```
#include <cms/MessageEOFException.h>
#include <cms/MessageFormatException.h>
```

### Data Structures

- class **cms::BytesMessage**

*A **BytesMessage** (p. 1079) object is used to send a message containing a stream of unsigned bytes.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.562 src/main/cms/Closeable.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::Closeable**

*Interface for a class that implements the close method.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.563 src/main/decaf/io/Closeable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```



## Data Structures

- class **decaf::io::Closeable**

*Interface for a class that implements the close method.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.564 src/main/cms/CMSEException.h File Reference

```
#include <string>
#include <vector>
#include <iostream>
#include <exception>
#include <cms/Config.h>
```

## Data Structures

- class **cms::CMSEException**

*CMS API Exception that is the base for all exceptions thrown from CMS classes.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.565 src/main/cms/CMSPProperties.h File Reference

```
#include <cms/Config.h>
#include <map>
#include <string>
#include <vector>
```

## Data Structures

- class **cms::CMSProperties**  
*Interface for a Java-like properties object.*

## Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.566 src/main/cms/CMSSecurityException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::CMSSecurityException**  
*This exception must be thrown when a provider rejects a user name/password submitted by a client.*

## Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.567 src/main/cms/Connection.h File Reference

```
#include <cms/Config.h>
#include <cms/Startable.h>
#include <cms/Stoppable.h>
#include <cms/Closeable.h>
#include <cms/Session.h>
#include <cms/ConnectionMetaData.h>
```

## Data Structures

- class **cms::Connection**

*The client's connection to its provider.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.568 src/main/cms/ConnectionFactory.h File Reference

```
#include <cms/Config.h>
#include <cms/Connection.h>
#include <cms/CMSException.h>
#include <string>
```

## Data Structures

- class **cms::ConnectionFactory**

*Defines the interface for a factory that creates connection objects, the **Connection** (p. 1296) objects returned implement the CMS **Connection** (p. 1296) interface and hide the CMS Provider specific implementation details behind that interface.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.569 src/main/cms/ConnectionMetaData.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::ConnectionMetaData**

A *ConnectionMetaData* (p. 1425) object provides information describing the *Connection* (p. 1296) object.

## Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

## 7.570 src/main/cms/DeliveryMode.h File Reference

```
#include <cms/Config.h>
```

## Data Structures

- class **cms::DeliveryMode**

This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.

## Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

## 7.571 src/main/cms/Destination.h File Reference

```
#include <cms/CMSProperties.h>
#include <cms/Config.h>
#include <string>
```

## Data Structures

- class **cms::Destination**

A *Destination* (p. 1776) object encapsulates a provider-specific address.

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.572 src/main/cms/ExceptionListener.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::ExceptionListener**

*If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 1893) that is registered with the **Connection** (p. 1296).*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.573 src/main/cms/IllegalStateException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::IllegalStateException**

*This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.574 src/main/decaf/lang/exceptions/IllegalStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::IllegalStateException**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.575 src/main/cms/InvalidClientIdException.h File Reference

```
#include <cms/Config.h>  
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::InvalidClientIdException**

*This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.576 src/main/cms/InvalidDestinationException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::InvalidDestinationException**

*This exception must be thrown when a destination either is not understood by a provider or is no longer valid.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.577 src/main/cms/InvalidSelectorException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::InvalidSelectorException**

*This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.578 src/main/cms/MapMessage.h File Reference

```
#include <cms/Config.h>
```

```
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotWriteableException.h>
```

## Data Structures

- class **cms::MapMessage**

*A **MapMessage** (p. 2551) object is used to send a set of name-value pairs.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.579 src/main/cms/MessageConsumer.h File Reference

```
#include <cms/Config.h>
#include <cms/MessageListener.h>
#include <cms/Message.h>
#include <cms/Closeable.h>
```

## Data Structures

- class **cms::MessageConsumer**

*A client uses a **MessageConsumer** (p. 2674) to received messages from a destination.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*



## 7.580 src/main/cms/MessageEnumeration.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::MessageEnumeration**  
*Defines an object that enumerates a collection of Messages.*

### Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.581 src/main/cms/MessageEOFException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::MessageEOFException**  
*This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 3760) or **BytesMessage** (p. 1079) is being read.*

### Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.582 src/main/cms/MessageFormatException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::MessageFormatException**

*This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.583 src/main/cms/MessageListener.h File Reference

```
#include <cms/Config.h>
```

## Data Structures

- class **cms::MessageListener**

*A **MessageListener** (p. 2779) object is used to receive asynchronously delivered messages.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.584 src/main/cms/MessageNotReadableException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::MessageNotReadableException**

*This exception must be thrown when a CMS client attempts to read a write-only message.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.585 src/main/cms/MessageNotWriteableException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::MessageNotWriteableException**

*This exception must be thrown when a CMS client attempts to write to a read-only message.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.586 src/main/cms/MessageProducer.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/Destination.h>
#include <cms/Closeable.h>
#include <cms/CMSException.h>
#include <cms/InvalidDestinationException.h>
#include <cms/MessageFormatException.h>
#include <cms/UnsupportedOperationException.h>
#include <cms/DeliveryMode.h>
```

## Data Structures

- class **cms::MessageProducer**

*A client uses a **MessageProducer** (p. 2809) object to send messages to a **Destination** (p. 1776).*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.587 src/main/cms/ObjectMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
```

## Data Structures

- class **cms::ObjectMessage**

*Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.588 src/main/cms/Queue.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::Queue**

*An interface encapsulating a provider-specific queue name.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.589 src/main/decaf/util/Queue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractCollection.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

## Data Structures

- class **decaf::util::Queue< E >**

*A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.590 src/main/cms/QueueBrowser.h File Reference

```
#include <string>
#include <cms/Config.h>
```

```
#include <cms/Closeable.h>
#include <cms/Queue.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageEnumeration.h>
```

## Data Structures

- class **cms::QueueBrowser**

*This class implements in interface for browsing the messages in a **Queue** (p.3238) without removing them.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.591 src/main/cms/Session.h File Reference

```
#include <cms/Config.h>
#include <cms/Closeable.h>
#include <cms/Message.h>
#include <cms/TextMessage.h>
#include <cms/BytesMessage.h>
#include <cms/MapMessage.h>
#include <cms/StreamMessage.h>
#include <cms/MessageProducer.h>
#include <cms/MessageConsumer.h>
#include <cms/Topic.h>
#include <cms/Queue.h>
#include <cms/QueueBrowser.h>
#include <cms/TemporaryTopic.h>
#include <cms/TemporaryQueue.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::Session**

A *Session* (p. 3460) object is a single-threaded context for producing and consuming messages.

## Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

## 7.592 src/main/cms/Startable.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::Startable**

Interface for a class that implements the start method.

## Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

## 7.593 src/main/cms/Stopable.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::Stopable**

Interface for a class that implements the stop method.

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.594 src/main/cms/StreamMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWriteableException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageEOFException.h>
```

## Data Structures

- class **cms::StreamMessage**

*Interface for a **StreamMessage** (p. 3760).*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.595 src/main/cms/TemporaryQueue.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::TemporaryQueue**

*Defines a Temporary **Queue** (p. 3238) based **Destination** (p. 1776).*



## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.596 src/main/cms/TemporaryTopic.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::TemporaryTopic**

*Defines a Temporary **Topic** (p. 3930) based **Destination** (p. 1776).*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.597 src/main/cms/TextMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageNotWriteableException.h>
```

## Data Structures

- class **cms::TextMessage**

*Interface for a text message.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.598 src/main/cms/Topic.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::Topic**

*An interface encapsulating a provider-specific topic name.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.599 src/main/cms/UnsupportedOperationException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::UnsupportedOperationException**

*This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.600 src/main/decaf/lang/exceptions/UnsupportedOperationException.h File Reference

```
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::lang::exceptions::UnsupportedOperationException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.601 src/main/decaf/internal/AprPool.h File Reference

```
#include <decaf/util/Config.h>
#include <apr_pools.h>
```

## Data Structures

- class **decaf::internal::AprPool**

*Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**

## 7.602 src/main/decaf/internal/DecafRuntime.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runtime.h>
#include <apr_pools.h>
```

### Data Structures

- class **decaf::internal::DecafRuntime**  
*Handles APR initialization and termination.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**

## 7.603 src/main/decaf/internal/io/StandardErrorOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
```

### Data Structures

- class **decaf::internal::io::StandardErrorOutputStream**  
*Wrapper Around the Standard error Output facility on the current platform.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::io**

## 7.604 src/main/decaf/internal/io/StandardInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
```

### Data Structures

- class **decaf::internal::io::StandardInputStream**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::io**

## 7.605 src/main/decaf/internal/io/StandardOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
```

### Data Structures

- class **decaf::internal::io::StandardOutputStream**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::io**

## 7.606 src/main/decaf/internal/net/DefaultServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ServerSocketFactory.h>
```

### Data Structures

- class **decaf::internal::net::DefaultServerSocketFactory**  
*Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**

## 7.607 src/main/decaf/internal/net/DefaultSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketFactory.h>
```

### Data Structures

- class **decaf::internal::net::DefaultSocketFactory**  
*SocketFactory implementation that is used to create Sockets.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**

## 7.608 src/main/decaf/internal/net/Network.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/internal/util/Resource.h>
#include <decaf/internal/util/GenericResource.h>
```

### Data Structures

- class **decaf::internal::net::Network**  
*Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**

## 7.609 src/main/decaf/internal/net/SocketFileDescriptor.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/FileDescriptor.h>
```

### Data Structures

- class **decaf::internal::net::SocketFileDescriptor**  
*File Descriptor type used internally by Decaf Socket objects.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**

## 7.610 src/main/decaf/internal/net/ssl/DefaultSSLContext.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLContext.h>
```

### Data Structures

- class **decaf::internal::net::ssl::DefaultSSLContext**

*Default SSLContext manager for the Decaf library.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**

## 7.611 src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLServerSocketFactory.h>
```

### Data Structures

- class **decaf::internal::net::ssl::DefaultSSLServerSocketFactory**

*Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*



## 7.612 src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h File Reference 1601

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**

## 7.612 src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLSocketFactory.h>
#include <string>
#include <vector>
```

### Data Structures

- class **decaf::internal::net::ssl::DefaultSSLSocketFactory**  
*Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**

## 7.613 src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLContextSpi.h>
```

### Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLContextSpi**  
*Provides an SSLContext that wraps the OpenSSL API.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.614 src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
#include <vector>
```

## Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLParameters**

*Container class for parameters that are Common to OpenSSL socket classes.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.615 src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLServerSocket.h>
```

## Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLServerSocket**

*SSLServerSocket based on OpenSSL library code.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.616 src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLServerSocketFactory.h>
```

## Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory**

*SSLServerSocketFactory that creates Server Sockets that use OpenSSL.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.617 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLSocket.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
```

### Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocket**

*Wraps a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.618 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

### Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketException**

*Subclass of the standard SocketException that knows how to produce an error message from the OpenSSL error stack.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.619 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLSocketFactory.h>
```

## Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory**  
*Client Socket Factory that creates SSL based client sockets using the OpenSSL library.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.620 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
```

## Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream**

*An output stream for reading data from an OpenSSL Socket instance.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.621 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
```

## Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream**

*OutputStream implementation used to write data to an **OpenSSLSocket** (p. 2940) instance.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.622 src/main/decaf/internal/net/tcp/TcpSocket.h File Reference

```
#include <decaf/net/SocketException.h>
#include <decaf/net/SocketImpl.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/util/Config.h>
#include <decaf/internal/AprPool.h>
#include <apr_network_io.h>
#include <decaf/io/IOException.h>
#include <decaf/net/SocketTimeoutException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **decaf::internal::net::tcp::TcpSocket**  
*Platform-independent implementation of the socket interface.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::tcp**

## 7.623 src/main/decaf/internal/net/tcp/TcpSocketInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

## Data Structures

- class **decaf::internal::net::tcp::TcpSocketInputStream**

*Input stream for performing reads on a socket.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::tcp**

## 7.624 src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
```

## Data Structures

- class **decaf::internal::net::tcp::TcpSocketOutputStream**

*Output stream for performing write operations on a socket.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::tcp**

## 7.625 src/main/decaf/internal/net/URLEncoderDecoder.h File Reference

```
#include <decaf/util/Config.h>
```



```
#include <decaf/net/URISyntaxException.h>
#include <string>
```

### Data Structures

- class **decaf::internal::net::URIEncoderDecoder**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**

## 7.626 src/main/decaf/internal/net/URIHelper.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
#include <decaf/net/URISyntaxException.h>
#include <decaf/internal/net/URIType.h>
```

### Data Structures

- class **decaf::internal::net::URIHelper**  
*Helper class used by the URI classes in encoding and decoding of URI's.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**

## 7.627 src/main/decaf/internal/net/URIType.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::internal::net::URIType**

*Basic type object that holds data that composes a given URI.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::net**

## 7.628 src/main/decaf/internal/nio/BufferFactory.h File Reference

```
#include <decaf/nio/ByteBuffer.h>
#include <decaf/nio/CharBuffer.h>
#include <decaf/nio/DoubleBuffer.h>
#include <decaf/nio/FloatBuffer.h>
#include <decaf/nio/LongBuffer.h>
#include <decaf/nio/IntBuffer.h>
#include <decaf/nio/ShortBuffer.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

## Data Structures

- class **decaf::internal::nio::BufferFactory**

*Factory class used by static methods in the **decaf::nio** (p. 147) package to create the various default version of the NIO interfaces.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

## 7.629 src/main/decaf/internal/nio/ByteArrayBuffer.h File Reference

```
#include <decaf/nio/ByteBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/nio/CharBuffer.h>
#include <decaf/nio/DoubleBuffer.h>
#include <decaf/nio/FloatBuffer.h>
#include <decaf/nio/ShortBuffer.h>
#include <decaf/nio/IntBuffer.h>
#include <decaf/nio/LongBuffer.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **decaf::internal::nio::ByteArrayBuffer**

*This class defines six categories of operations upon byte buffers:*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

## 7.630 src/main/decaf/internal/nio/CharArrayBuffer.h File Reference

```
#include <decaf/nio/CharBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

```
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **decaf::internal::nio::CharArrayBuffer**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

## 7.631 src/main/decaf/internal/nio/DoubleArrayBuffer.h File Reference

```
#include <decaf/nio/DoubleBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **decaf::internal::nio::DoubleArrayBuffer**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

## 7.632 src/main/decaf/internal/nio/FloatArrayBuffer.h File Reference

```
#include <decaf/nio/FloatBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **decaf::internal::nio::FloatArrayBuffer**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

## 7.633 src/main/decaf/internal/nio/IntArrayBuffer.h File Reference

```
#include <decaf/nio/IntBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **decaf::internal::nio::IntArrayBuffer**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

## 7.634 src/main/decaf/internal/nio/LongArrayBuffer.h File Reference

```
#include <decaf/nio/LongBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **decaf::internal::nio::LongArrayBuffer**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

## 7.635 src/main/decaf/internal/nio/ShortArrayBuffer.h File Reference

```
#include <decaf/nio/ShortBuffer.h>
```

## 7.636 src/main/decaf/internal/security/unix/SecureRandomImpl.h File Reference

```
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **decaf::internal::nio::ShortArrayBuffer**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

## 7.636 src/main/decaf/internal/security/unix/SecureRandomImpl.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/SecureRandomSpi.h>
```

### Data Structures

- class **decaf::internal::security::SecureRandomImpl**  
*Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::security**

## 7.637 src/main/decaf/internal/security/windows/SecureRandomImpl.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/SecureRandomSpi.h>
```

### Data Structures

- class **decaf::internal::security::SecureRandomImpl**

*Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::security**

## 7.638 src/main/decaf/internal/util/ByteArrayAdapter.h File Reference

```
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
```

### Data Structures

- class **decaf::internal::util::ByteArrayAdapter**

*This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.*

- union **decaf::internal::util::ByteArrayAdapter::Array**



## **7.639 src/main/decaf/internal/util/concurrent/ConditionImpl.h File Reference**

### **Namespaces**

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::util**

## **7.639 src/main/decaf/internal/util/concurrent/ConditionImpl.h File Reference**

```
#include <decaf/util/Config.h>
```

### **Data Structures**

- class **decaf::internal::util::concurrent::ConditionImpl**

### **Namespaces**

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

## **7.640 src/main/decaf/internal/util/concurrent/MutexImpl.h File Reference**

```
#include <decaf/util/Config.h>
```

### **Data Structures**

- class **decaf::internal::util::concurrent::MutexImpl**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

### 7.641 src/main/decaf/internal/util/concurrent/SynchronizableImpl.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
```

## Data Structures

- class **decaf::internal::util::concurrent::SynchronizableImpl**

*A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

### 7.642 src/main/decaf/internal/util/concurrent/Transferer.h File Reference

```
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/util/concurrent/TimeoutException.h>
```

## Data Structures

- class **decaf::internal::util::concurrent::Transferer**< E >

*Shared internal API for dual stacks and queues.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

## 7.643 src/main/decaf/internal/util/concurrent/TransferQueue.h File Reference

```
#include <decaf/internal/util/concurrent/Transferer.h>
#include <decaf/util/concurrent/locks/LockSupport.h>
#include <decaf/util/concurrent/atomic/AtomicReference.h>
#include <decaf/util/concurrent/TimeoutException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/Thread.h>
```

## Data Structures

- class **decaf::internal::util::concurrent::TransferQueue**< E >

*This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

## 7.644 src/main/decaf/internal/util/concurrent/TransferStack.h File Reference

```
#include <decaf/internal/util/concurrent/Transferer.h>
#include <decaf/util/concurrent/TimeoutException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
```

### Data Structures

- class **decaf::internal::util::concurrent::TransferStack**< E >

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

## 7.645 src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::ConditionHandle**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.646 src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::ConditionHandle**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.647 src/main/decaf/internal/util/concurrent/unix/MutexHandle.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::MutexHandle**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.648 src/main/decaf/internal/util/concurrent/windows/MutexHandle.h File Reference

```
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::util::concurrent::MutexHandle**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.649 src/main/decaf/internal/util/GenericResource.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/internal/util/Resource.h>
```

## Data Structures

- class **decaf::internal::util::GenericResource< T >**  
*A Generic **Resource** (p. 3374) wraps some type and will delete it when the **Resource** (p. 3374) itself is deleted.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::util**

## 7.650 src/main/decaf/internal/util/HexStringParser.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
```

## Data Structures

- class **decaf::internal::util::HexStringParser**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::util**

## 7.651 src/main/decaf/internal/util/Resource.h File Reference

```
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::internal::util::Resource**

*Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::util**

## 7.652 src/main/decaf/internal/util/TimerTaskHeap.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/TimerTask.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **decaf::internal::util::TimerTaskHeap**

*A Binary Heap implemented specifically for the Timer class in Decaf Util.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::util**

## 7.653 src/main/decaf/internal/util/zip/crc32.h File Reference

### Variables

- local const unsigned long FAR **crc\_table** [TBLS][256]

### 7.653.1 Variable Documentation

7.653.1.1 local const unsigned long FAR **crc\_table**[TBLS][256]

## 7.654 src/main/decaf/internal/util/zip/deflate.h File Reference

```
#include "zutil.h"
```

### Data Structures

- struct **ct\_data\_s**
- struct **tree\_desc\_s**
- struct **internal\_state**

### Defines

- #define **GZIP**
- #define **LENGTH\_CODES** 29
- #define **LITERALS** 256
- #define **L\_CODES** (LITERALS+1+LENGTH\_CODES)
- #define **D\_CODES** 30
- #define **BL\_CODES** 19
- #define **HEAP\_SIZE** (2\*L\_CODES+1)
- #define **MAX\_BITS** 15
- #define **INIT\_STATE** 42
- #define **EXTRA\_STATE** 69
- #define **NAME\_STATE** 73
- #define **COMMENT\_STATE** 91
- #define **HCRC\_STATE** 103



- `#define BUSY_STATE 113`
- `#define FINISH_STATE 666`
- `#define Freq fc.freq`
- `#define Code fc.code`
- `#define Dad dl.dad`
- `#define Len dl.len`
- `#define max_insert_length max_lazy_match`
- `#define put_byte(s, c) { s->pending_buf[s->pending++] = (c); }`
- `#define MIN_LOOKAHEAD (MAX_MATCH+MIN_MATCH+1)`
- `#define MAX_DIST(s) ((s)->w_size-MIN_LOOKAHEAD)`
- `#define WIN_INIT MAX_MATCH`
- `#define d_code(dist) ((dist) < 256 ? _dist_code[dist] : _dist_code[256+((dist)>>7)])`
- `#define _tr_tally_lit(s, c, flush)`
- `#define _tr_tally_dist(s, distance, length, flush)`

## Typedefs

- `typedef struct ct_data_s ct_data`
- `typedef struct static_tree_desc_s static_tree_desc`
- `typedef struct tree_desc_s tree_desc`
- `typedef ush Pos`
- `typedef Pos FAR Posf`
- `typedef unsigned IPos`
- `typedef struct internal_state deflate_state`

## Functions

- `void ZLIB_INTERNAL _tr_init OF ((deflate_state *s))`
- `int ZLIB_INTERNAL _tr_tally OF ((deflate_state *s, unsigned dist, unsigned lc))`
- `void ZLIB_INTERNAL _tr_flush_block OF ((deflate_state *s, charf *buf, ulg stored_len, int last))`

## Variables

- `uch ZLIB_INTERNAL _length_code [ ]`
- `uch ZLIB_INTERNAL _dist_code [ ]`

### 7.654.1 Define Documentation

#### 7.654.1.1 `#define _tr_tally_dist( s, distance, length, flush )`

**Value:**

```
{ uch len = (length); \
  ush dist = (distance); \
  s->d_buf[s->last_lit] = dist; \
  s->l_buf[s->last_lit++] = len; \
  dist--; \
  s->dyn_ltree[_length_code[len]+LITERALS+1].Freq++; \
  s->dyn_dtree[d_code(dist)].Freq++; \
  flush = (s->last_lit == s->lit_bufsize-1); \
}
```

#### 7.654.1.2 `#define tr_tally_lit( s, c, flush )`

**Value:**

```
{ uch cc = (c); \
  s->d_buf[s->last_lit] = 0; \
  s->l_buf[s->last_lit++] = cc; \
  s->dyn_ltree[cc].Freq++; \
  flush = (s->last_lit == s->lit_bufsize-1); \
}
```

7.654.1.3 `#define BL_CODES 19`

7.654.1.4 `#define BUSY_STATE 113`

7.654.1.5 `#define Code fc.code`

7.654.1.6 `#define COMMENT_STATE 91`

7.654.1.7 `#define d_code( dist ) ((dist) < 256 ? _dist_code[dist] :  
_dist_code[256+((dist)>>7)])`

7.654.1.8 `#define D_CODES 30`

7.654.1.9 `#define Dad dl.dad`

7.654.1.10 `#define EXTRA_STATE 69`

7.654.1.11 `#define FINISH_STATE 666`

7.654.1.12 `#define Freq fc.freq`

7.654.1.13 `#define GZIP`

7.654.1.14 `#define HCRC_STATE 103`

7.654.1.15 `#define HEAP_SIZE (2*L_CODES+1)`

7.654.1.16 `#define INIT_STATE 42`

7.654.1.17 `#define L_CODES (LITERALS+1+LENGTH_CODES)`

7.654.1.18 `#define Len dl.len`

7.654.1.19 `#define LENGTH_CODES 29`

7.654.1.20 `#define LITERALS 256`

7.654.1.21 `#define MAX_BITS 15`

7.654.1.22 `#define MAX_DIST( s ) ((s)->w_size-MIN_LOOKAHEAD)`

7.654.1.23 `#define max_insert_length max_lazy_match`

7.654.1.24 `#define MIN_LOOKAHEAD (MAX_MATCH+MIN_MATCH+1)`

7.654.1.25 `#define NAME_STATE 73`

7.654.1.26 `#define put_byte( s, c ) {s->pending_buf[s->pending++] = (c);}`

7.654.1.27 `#define WIN_INIT MAX_MATCH`

---

Generated on Tue Mar 8 2011 17:10:05 for activemq-cpp-3.2.5 by Doxygen

## 7.654.2 Typedef Documentation

7.654.2.1 `typedef struct ct_data_s ct_data`

7.654.2.2 `typedef struct internal_state deflate_state`

7.654.2.3 `typedef unsigned IPos`

```
#include "zlib.h"
#include <fcntl.h>
```

## Data Structures

- struct **gz\_state**

## Defines

- #define **ZLIB\_INTERNAL**
- #define **local** static
- #define **zstrerror()** "stdio error (consult errno)"
- #define **GZBUFSIZE** 8192
- #define **GZ\_NONE** 0
- #define **GZ\_READ** 7247
- #define **GZ\_WRITE** 31153
- #define **GZ\_APPEND** 1
- #define **LOOK** 0
- #define **COPY** 1
- #define **GZIP** 2
- #define **GT\_OFF(x)** (sizeof(int) == sizeof(z\_off64\_t) && (x) > gz\_intmax())

## Typedefs

- typedef **gz\_state FAR \* gz\_statep**

## Functions

- **voidp** malloc **OF** ((**uInt** size))
- void free **OF** ((**voidpf** ptr))
- ZEXTERN **gzFile** ZEXPORT gzopen64 **OF** ((const char \*, const char \*))
- ZEXTERN z\_off64\_t ZEXPORT gzseek64 **OF** ((**gzFile**, z\_off64\_t, int))
- ZEXTERN z\_off64\_t ZEXPORT gztell64 **OF** ((**gzFile**))
- void ZLIB\_INTERNAL gz\_error **OF** ((**gz\_statep**, int, const char \*))
- unsigned ZLIB\_INTERNAL gz\_intmax **OF** ((void))

### 7.655.1 Define Documentation

7.655.1.1 `#define COPY 1`

7.655.1.2 `#define GT_OFF( x ) (sizeof(int) == sizeof(z_off64_t) && (x) > gz_intmax())`

7.655.1.3 `#define GZ_APPEND 1`

7.655.1.4 `#define GZ_NONE 0`

7.655.1.5 `#define GZ_READ 7247`

7.655.1.6 `#define GZ_WRITE 31153`

7.655.1.7 `#define GZBUFSIZE 8192`

7.655.1.8 `#define GZIP 2`

7.655.1.9 `#define local static`

7.655.1.10 `#define LOOK 0`

7.655.1.11 `#define ZLIB_INTERNAL`

7.655.1.12 `#define zstrerror( ) "stdio error (consult errno)"`

### 7.655.2 Typedef Documentation

7.655.2.1 `typedef gz_state FAR* gz_statep`

### 7.655.3 Function Documentation

7.655.3.1 `voidp malloc OF ( (uInt size) )`

7.655.3.2 `unsigned ZLIB_INTERNAL gz_intmax OF ( (void) )`

7.655.3.3 `void ZLIB_INTERNAL gz_error OF ( (gz_statep, int, const char *) )`

7.655.3.4 `ZEXTERN z_off64_t ZEXPORT gztell64 OF ( (gzFile) )`

7.655.3.5 `ZEXTERN z_off64_t ZEXPORT gzseek64 OF ( (gzFile, z_off64_t, int) )`

7.655.3.6 `ZEXTERN gzFile ZEXPORT gzopen64 OF ( (const char *, const char *) )`

7.655.3.7 `void free OF ( (voidpf ptr) )`

## 7.656 src/main/decaf/internal/util/zip/inffast.h File Reference

### Functions

---

Generated on Tue Mar 8 2011 17:10:05 for activemq-cpp-3.2.5 by Doxygen

- `void ZLIB_INTERNAL inflate_fast OF ((z_streamp strm, unsigned start))`

### 7.656.1 Function Documentation

7.656.1.1 void ZLIB\_INTERNAL inflate\_fast OF ( (z\_streamp strm, unsigned start) )

### 7.657 src/main/decaf/internal/util/zip/inffixed.h File Reference

### 7.658 src/main/decaf/internal/util/zip/inflate.h File Reference

#### Data Structures

- struct **inflate\_state**

#### Defines

- #define **GUNZIP**

#### Enumerations

- enum **inflate\_mode** {  
    **HEAD, FLAGS, TIME, OS,**  
    **EXLEN, EXTRA, NAME, COMMENT,**  
    **HCRC, DICTID, DICT, TYPE,**  
    **TYPEDO, STORED, COPY\_, COPY,**  
    **TABLE, LENLENS, CODELENS, LEN\_,**  
    **LEN, LENEXT, DIST, DISTEXT,**  
    **MATCH, LIT, CHECK, LENGTH,**  
    **DONE, BAD, MEM, SYNC }**

### 7.658.1 Define Documentation

7.658.1.1 #define **GUNZIP**

### 7.658.2 Enumeration Type Documentation

7.658.2.1 enum **inflate\_mode**

#### Enumerator:

***HEAD***  
***FLAGS***  
***TIME***  
***OS***

*EXLEN*  
*EXTRA*  
*NAME*  
*COMMENT*  
*HCRC*  
*DICTID*  
*DICT*  
*TYPE*  
*TYPEDO*  
*STORED*  
*COPY\_*  
*COPY*  
*TABLE*  
*LENLENS*  
*CODELENS*  
*LEN\_*  
*LEN*  
*LENEXT*  
*DIST*  
*DISTEXT*  
*MATCH*  
*LIT*  
*CHECK*  
*LENGTH*  
*DONE*  
*BAD*  
*MEM*  
*SYNC*

## 7.659 src/main/decaf/internal/util/zip/inftrees.h File Reference

### Data Structures

- struct `code`

### Defines

- #define **ENOUGH\_LENS** 852
- #define **ENOUGH\_DISTS** 592
- #define **ENOUGH** (ENOUGH\_LENS+ENOUGH\_DISTS)

## Enumerations

- enum **codetype** { **CODES**, **LENS**, **DISTS** }

## Functions

- int ZLIB\_INTERNAL inflate\_table **OF** ((**codetype** type, unsigned short FAR \*lens, unsigned codes, **code** FAR \*FAR \*table, unsigned FAR \*bits, unsigned short FAR \*work))

### 7.659.1 Define Documentation

7.659.1.1 **#define** ENOUGH (ENOUGH\_LENS+ENOUGH\_DISTS)

7.659.1.2 **#define** ENOUGH\_DISTS 592

7.659.1.3 **#define** ENOUGH\_LENS 852

### 7.659.2 Enumeration Type Documentation

7.659.2.1 enum **codetype**

Enumerator:

***CODES***

***LENS***

***DISTS***

### 7.659.3 Function Documentation

7.659.3.1 int ZLIB\_INTERNAL inflate\_table **OF** ( (**codetype** type, unsigned short FAR \*lens, unsigned codes, **code** FAR \*FAR \*table, unsigned FAR \*bits, unsigned short FAR \*work) )

## 7.660 src/main/decaf/internal/util/zip/trees.h File Reference

### Variables

- local const **ct\_data** static\_ltree [L\_CODES+2]
- local const **ct\_data** static\_dtree [D\_CODES]
- const **uch** ZLIB\_INTERNAL \_dist\_code [DIST\_CODE\_LEN]
- const **uch** ZLIB\_INTERNAL \_length\_code [MAX\_MATCH-MIN\_MATCH+1]
- local const int **base\_length** [LENGTH\_CODES]
- local const int **base\_dist** [D\_CODES]



**7.660.1.1** `const uch ZLIB_INTERNAL _dist_code[DIST_CODE_LEN]`

[illegible]

```

{
0, 1, 2, 3, 4, 5, 6, 7, 8, 8, 9, 9, 10, 10, 11, 11, 12, 12, 12, 12,
13, 13, 13, 13, 14, 14, 14, 14, 15, 15, 15, 15, 16, 16, 16, 16, 16, 16, 16,
17, 17, 17, 17, 17, 17, 17, 17, 18, 18, 18, 18, 18, 18, 18, 18, 19, 19, 19, 19,
19, 19, 19, 19, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 22, 22, 22, 22,
22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 23, 23, 23, 23, 23, 23, 23,
23, 23, 23, 23, 23, 23, 23, 23, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
}

```

**7.660.1.3 local const int base\_dist[D\_CODES]****Initial value:**

```
{
    0,      1,      2,      3,      4,      6,      8,      12,      16,      24,
   32,     48,     64,     96,    128,    192,    256,    384,    512,    768,
 1024,  1536,  2048,  3072,  4096,  6144,  8192, 12288, 16384, 24576
}
```

**7.660.1.4 local const int base\_length[LENGTH\_CODES]****Initial value:**

```
{
0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 28, 32, 40, 48, 56,
64, 80, 96, 112, 128, 160, 192, 224, 0
}
```

**7.660.1.5 local const ct\_data static\_dtree[D\_CODES]****Initial value:**

```
{
{{ 0},{ 5}}, {{16},{ 5}}, {{ 8},{ 5}}, {{24},{ 5}}, {{ 4},{ 5}},
{{20},{ 5}}, {{12},{ 5}}, {{28},{ 5}}, {{ 2},{ 5}}, {{18},{ 5}},
{{10},{ 5}}, {{26},{ 5}}, {{ 6},{ 5}}, {{22},{ 5}}, {{14},{ 5}},
{{30},{ 5}}, {{ 1},{ 5}}, {{17},{ 5}}, {{ 9},{ 5}}, {{25},{ 5}},
{{ 5},{ 5}}, {{21},{ 5}}, {{13},{ 5}}, {{29},{ 5}}, {{ 3},{ 5}},
{{19},{ 5}}, {{11},{ 5}}, {{27},{ 5}}, {{ 7},{ 5}}, {{23},{ 5}}
}
```

**7.660.1.6 local const ct\_data static\_ltree[L\_CODES+2]****7.661 src/main/decaf/internal/util/zip/zconf.h File Reference**

```
#include <decaf/util/Config.h>
```

**Defines**

- #define **const**
- #define **MAX\_MEM\_LEVEL** 9
- #define **MAX\_WBITS** 15
- #define **OF**(args) ()
- #define **ZEXTERN** extern
- #define **ZEXPORT**
- #define **ZEXPORTVA**

- `#define FAR`
- `#define SEEK_SET 0`
- `#define SEEK_CUR 1`
- `#define SEEK_END 2`
- `#define z_off_t long`
- `#define z_off64_t z_off_t`

## Typedefs

- `typedef unsigned char Byte`
- `typedef unsigned int uInt`
- `typedef unsigned long uLong`
- `typedef Byte FAR Bytef`
- `typedef char FAR charf`
- `typedef int FAR intf`
- `typedef uInt FAR uIntf`
- `typedef uLong FAR uLongf`
- `typedef Byte const * voidpc`
- `typedef Byte FAR * voidpf`
- `typedef Byte * voidp`

**7.661.1 Define Documentation**7.661.1.1 **#define** const7.661.1.2 **#define** FAR7.661.1.3 **#define** MAX\_MEM\_LEVEL 97.661.1.4 **#define** MAX\_WBITS 157.661.1.5 **#define** OF( *args* )()7.661.1.6 **#define** SEEK\_CUR 17.661.1.7 **#define** SEEK\_END 27.661.1.8 **#define** SEEK\_SET 07.661.1.9 **#define** z\_off64\_t z\_off\_t7.661.1.10 **#define** z\_off\_t long7.661.1.11 **#define** ZEXPORT7.661.1.12 **#define** ZEXPORTVA7.661.1.13 **#define** ZEXTERN extern**7.661.2 Typedef Documentation**7.661.2.1 **typedef** unsigned char Byte7.661.2.2 **typedef** Byte FAR Bytef7.661.2.3 **typedef** char FAR charf7.661.2.4 **typedef** int FAR intf7.661.2.5 **typedef** unsigned int uInt7.661.2.6 **typedef** uInt FAR uIntf7.661.2.7 **typedef** unsigned long uLong7.661.2.8 **typedef** uLong FAR uLongf7.661.2.9 **typedef** Byte\* voidp7.661.2.10 **typedef** Byte const\* voidpc7.661.2.11 **typedef** Byte FAR\* voidpf

---

Generated on Tue Mar 8 2011 17:10:05 for activemq-cpp-3.2.5 by Doxygen**7.662 src/main/decaf/internal/util/zip/zlib.h File Reference**

```
#include "zconf.h"
```

## Data Structures

- struct **z\_stream\_s**
- struct **gz\_header\_s**
- struct **internal\_state**

## Defines

- #define **ZLIB\_VERSION** "1.2.5"
- #define **ZLIB\_VERNUM** 0x1250
- #define **ZLIB\_VER\_MAJOR** 1
- #define **ZLIB\_VER\_MINOR** 2
- #define **ZLIB\_VER\_REVISION** 5
- #define **ZLIB\_VER\_SUBREVISION** 0
- #define **Z\_NO\_FLUSH** 0
- #define **Z\_PARTIAL\_FLUSH** 1
- #define **Z\_SYNC\_FLUSH** 2
- #define **Z\_FULL\_FLUSH** 3
- #define **Z\_FINISH** 4
- #define **Z\_BLOCK** 5
- #define **Z\_TREES** 6
- #define **Z\_OK** 0
- #define **Z\_STREAM\_END** 1
- #define **Z\_NEED\_DICT** 2
- #define **Z\_ERRNO** (-1)
- #define **Z\_STREAM\_ERROR** (-2)
- #define **Z\_DATA\_ERROR** (-3)
- #define **Z\_MEM\_ERROR** (-4)
- #define **Z\_BUF\_ERROR** (-5)
- #define **Z\_VERSION\_ERROR** (-6)
- #define **Z\_NO\_COMPRESSION** 0
- #define **Z\_BEST\_SPEED** 1
- #define **Z\_BEST\_COMPRESSION** 9
- #define **Z\_DEFAULT\_COMPRESSION** (-1)
- #define **Z\_FILTERED** 1
- #define **Z\_HUFFMAN\_ONLY** 2
- #define **Z\_RLE** 3
- #define **Z\_FIXED** 4
- #define **Z\_DEFAULT\_STRATEGY** 0
- #define **Z\_BINARY** 0
- #define **Z\_TEXT** 1
- #define **Z\_ASCII** Z\_TEXT
- #define **Z\_UNKNOWN** 2
- #define **Z\_DEFLATED** 8
- #define **Z\_NULL** 0
- #define **zlib\_version** zlibVersion()

- #define **deflateInit**(strm, level) deflateInit\_((strm), (level), ZLIB\_VERSION, sizeof(**z\_stream**))
- #define **inflateInit**(strm) inflateInit\_((strm), ZLIB\_VERSION, sizeof(**z\_stream**))
- #define **deflateInit2**(strm, level, method, windowBits, memLevel, strategy)
- #define **inflateInit2**(strm, windowBits) inflateInit2\_((strm), (windowBits), ZLIB\_VERSION, sizeof(**z\_stream**))
- #define **inflateBackInit**(strm, windowBits, window)

## Typedefs

- typedef **voidpf** alloc\_func **OF** ((**voidpf** opaque, **uInt** items, **uInt** size))
- typedef struct **z\_stream\_s** **z\_stream**
- typedef **z\_stream** FAR \* **z\_streamp**
- typedef struct **gz\_header\_s** **gz\_header**
- typedef **gz\_header** FAR \* **gz\_headerp**
- typedef **voidp** **gzFile**

## Functions

- ZEXTERN const char \*ZEXPORT zlibVersion **OF** ((void))
- ZEXTERN int ZEXPORT deflate **OF** ((**z\_streamp** strm, int flush))
- ZEXTERN int ZEXPORT deflateEnd **OF** ((**z\_streamp** strm))
- ZEXTERN int ZEXPORT deflateSetDictionary **OF** ((**z\_streamp** strm, const **Bytef** \*dictionary, uIntdictLength))
- ZEXTERN int ZEXPORT deflateCopy **OF** ((**z\_streamp** dest, **z\_streamp** source))
- ZEXTERN int ZEXPORT deflateParams **OF** ((**z\_streamp** strm, int level, int strategy))
- ZEXTERN int ZEXPORT deflateTune **OF** ((**z\_streamp** strm, int good\_length, int max\_lazy, int nice\_length, int max\_chain))
- ZEXTERN **uLong** ZEXPORT deflateBound **OF** ((**z\_streamp** strm, **uLong** sourceLen))
- ZEXTERN int ZEXPORT deflatePrime **OF** ((**z\_streamp** strm, int bits, int value))
- ZEXTERN int ZEXPORT deflateSetHeader **OF** ((**z\_streamp** strm, **gz\_headerp** head))
- ZEXTERN int ZEXPORT inflateReset2 **OF** ((**z\_streamp** strm, int windowBits))
- ZEXTERN int ZEXPORT inflateBack **OF** ((**z\_streamp** strm, in\_func in, void FAR \*in\_desc, out\_func out, void FAR \*out\_desc))
- ZEXTERN int ZEXPORT compress **OF** ((**Bytef** \*dest, **uLongf** \*destLen, const **Bytef** \*source, **uLong** sourceLen))
- ZEXTERN int ZEXPORT compress2 **OF** ((**Bytef** \*dest, **uLongf** \*destLen, const **Bytef** \*source, **uLong** sourceLen, int level))
- ZEXTERN **uLong** ZEXPORT compressBound **OF** ((**uLong** sourceLen))
- ZEXTERN **gzFile** ZEXPORT gzdopen **OF** ((int fd, const char \*mode))
- ZEXTERN int ZEXPORT gzbuffer **OF** ((**gzFile** file, unsigned size))
- ZEXTERN int ZEXPORT gzsetparams **OF** ((**gzFile** file, int level, int strategy))
- ZEXTERN int ZEXPORT gzread **OF** ((**gzFile** file, **voidp** buf, unsigned len))

- ZEXTERN int ZEXPORT gzwrite **OF** ((**gzFile** file, **voidpc** buf, unsigned len))
- ZEXTERN int ZEXPORTVA gzprintf **OF** ((**gzFile** file, const char \*format,...))
- ZEXTERN int ZEXPORT gzputs **OF** ((**gzFile** file, const char \*s))
- ZEXTERN char \*ZEXPORT gzgets **OF** ((**gzFile** file, char \*buf, int len))
- ZEXTERN int ZEXPORT gzputc **OF** ((**gzFile** file, int c))
- ZEXTERN int ZEXPORT gzgetc **OF** ((**gzFile** file))
- ZEXTERN int ZEXPORT gzungetc **OF** ((int c, **gzFile** file))
- ZEXTERN int ZEXPORT gzflush **OF** ((**gzFile** file, int flush))
- ZEXTERN const char \*ZEXPORT gzerror **OF** ((**gzFile** file, int \*errnum))
- ZEXTERN **uLong** ZEXPORT Adler32 **OF** ((**uLong** Adler, const **Bytef** \*buf, **uInt** len))
- ZEXTERN **uLong** ZEXPORT Crc32 **OF** ((**uLong** Crc, const **Bytef** \*buf, **uInt** len))
- ZEXTERN int ZEXPORT deflateInit\_ **OF** ((**z\_streamp** strm, int level, const char \*version, int stream\_size))
- ZEXTERN int ZEXPORT inflateInit\_ **OF** ((**z\_streamp** strm, const char \*version, int stream\_size))
- ZEXTERN int ZEXPORT deflateInit2\_ **OF** ((**z\_streamp** strm, intlevel, intmethod, int windowBits, int memLevel, int strategy, const char \*version, int stream\_size))
- ZEXTERN int ZEXPORT inflateInit2\_ **OF** ((**z\_streamp** strm, intwindowBits, const char \*version, int stream\_size))
- ZEXTERN int ZEXPORT inflateBackInit\_ **OF** ((**z\_streamp** strm, int windowBits, unsigned char FAR \*window, const char \*version, int stream\_size))
- ZEXTERN **gzFile** ZEXPORT gzopen **OF** ((const char \*, const char \*))
- ZEXTERN **z\_off\_t** ZEXPORT gzseek **OF** ((**gzFile**, **z\_off\_t**, int))
- ZEXTERN **z\_off\_t** ZEXPORT gztell **OF** ((**gzFile**))
- ZEXTERN **uLong** ZEXPORT Adler32\_combine **OF** ((**uLong**, **uLong**, **z\_off\_t**))
- ZEXTERN const char \*ZEXPORT zError **OF** ((int))
- ZEXTERN int ZEXPORT inflateSyncPoint **OF** ((**z\_streamp**))
- ZEXTERN int ZEXPORT inflateUndermine **OF** ((**z\_streamp**, int))

### 7.662.1 Define Documentation

7.662.1.1 **#define** deflateInit( *strm*, *level* ) deflateInit\_((strm), (level), ZLIB\_VERSION, sizeof(z\_stream))

7.662.1.2 **#define** deflateInit2( *strm*, *level*, *method*, *windowBits*, *memLevel*, *strategy* )

**Value:**

```
deflateInit2_((strm), (level), (method), (windowBits), (memLevel), \
              (strategy), ZLIB_VERSION, sizeof(z_stream))
```

7.662.1.3 **#define** inflateBackInit( *strm*, *windowBits*, *window* )

**Value:**

```
inflateBackInit_((strm), (windowBits), (window), \
                 ZLIB_VERSION, sizeof(z_stream))
```

7.662.1.4 `#define inflateInit( strm ) inflateInit_((strm), ZLIB_VERSION, sizeof(z_stream))`

7.662.1.5 `#define inflateInit2( strm, windowBits ) inflateInit2_((strm), (windowBits), ZLIB_VERSION, sizeof(z_stream))`

7.662.1.6 `#define Z_ASCII Z_TEXT`

7.662.1.7 `#define Z_BEST_COMPRESSION 9`

7.662.1.8 `#define Z_BEST_SPEED 1`

7.662.1.9 `#define Z_BINARY 0`

7.662.1.10 `#define Z_BLOCK 5`

7.662.1.11 `#define Z_BUF_ERROR (-5)`

7.662.1.12 `#define Z_DATA_ERROR (-3)`

7.662.1.13 `#define Z_DEFAULT_COMPRESSION (-1)`

7.662.1.14 `#define Z_DEFAULT_STRATEGY 0`

7.662.1.15 `#define Z_DEFLATED 8`

7.662.1.16 `#define Z_ERRNO (-1)`

7.662.1.17 `#define Z_FILTERED 1`

7.662.1.18 `#define Z_FINISH 4`

7.662.1.19 `#define Z_FIXED 4`

7.662.1.20 `#define Z_FULL_FLUSH 3`

7.662.1.21 `#define Z_HUFFMAN_ONLY 2`

7.662.1.22 `#define Z_MEM_ERROR (-4)`

7.662.1.23 `#define Z_NEED_DICT 2`

7.662.1.24 `#define Z_NO_COMPRESSION 0`

7.662.1.25 `#define Z_NO_FLUSH 0`

7.662.1.26 `#define Z_NULL 0`

7.662.1.27 `#define Z_OK 0`

7.662.1.28 `#define Z_PARTIAL_FLUSH 1`

---

Generated on Tue Mar 8 2011 17:10:05 for activemq-cpp-3.2.5 by Doxygen

7.662.1.29 `#define Z_RLE 3`

7.662.1.30 `#define Z_STREAM_END 1`

7.662.1.31 `#define Z_STREAM_ERROR (-2)`

7.662.1.32 `#define Z_SYNC_FLUSH 2`



## Defines

- `#define ZLIB_INTERNAL`
- `#define local static`
- `#define ERR_MSG(err) z_errmsg[Z_NEED_DICT-(err)]`
- `#define ERR_RETURN(strm, err) return (strm->msg = (char*)ERR_MSG(err), (err))`
- `#define DEF_WBITS MAX_WBITS`
- `#define DEF_MEM_LEVEL 8`
- `#define STORED_BLOCK 0`
- `#define STATIC_TREES 1`
- `#define DYN_TREES 2`
- `#define MIN_MATCH 3`
- `#define MAX_MATCH 258`
- `#define PRESET_DICT 0x20`
- `#define OS_CODE 0x03`
- `#define F_OPEN(name, mode) fopen((name), (mode))`
- `#define Assert(cond, msg)`
- `#define Trace(x)`
- `#define Tracev(x)`
- `#define Tracevv(x)`
- `#define Tracec(c, x)`
- `#define Tracecv(c, x)`
- `#define ZALLOC(strm, items, size) (*((strm)->zalloc))((strm)->opaque, (items), (size))`
- `#define ZFREE(strm, addr) (*((strm)->zfree))((strm)->opaque, (voidpf)(addr))`
- `#define TRY_FREE(s, p) {if (p) ZFREE(s, p);}`

## Typedefs

- `typedef unsigned char uch`
- `typedef uch FAR uchf`
- `typedef unsigned short ush`
- `typedef ush FAR ushf`
- `typedef unsigned long ulg`

## Functions

- `ZEXTERN uLong ZEXPORT Adler32_combine64 OF ((uLong, uLong, z_off_t))`
- `void ZLIB_INTERNAL zmemcpy OF ((Bytef *dest, const Bytef *source, uInt len))`
- `int ZLIB_INTERNAL zmemcmp OF ((const Bytef *s1, const Bytef *s2, uInt len))`
- `void ZLIB_INTERNAL zmemzero OF ((Bytef *dest, uInt len))`
- `voidpf ZLIB_INTERNAL zcalloc OF ((voidpf opaque, unsigned items, unsigned size))`
- `void ZLIB_INTERNAL zcfree OF ((voidpf opaque, voidpf ptr))`

## Variables

- `const char *const z_errmsg [10]`

### 7.663.1 Define Documentation

7.663.1.1 `#define Assert( cond, msg )`

7.663.1.2 `#define DEF_MEM_LEVEL 8`

7.663.1.3 `#define DEF_WBITS MAX_WBITS`

7.663.1.4 `#define DYN_TREES 2`

7.663.1.5 `#define ERR_MSG( err ) z_errmsg[Z_NEED_DICT-(err)]`

7.663.1.6 `#define ERR_RETURN( strm, err ) return (strm->msg = (char*)ERR_MSG(err), (err))`

7.663.1.7 `#define F_OPEN( name, mode ) fopen((name), (mode))`

7.663.1.8 `#define local static`

7.663.1.9 `#define MAX_MATCH 258`

7.663.1.10 `#define MIN_MATCH 3`

7.663.1.11 `#define OS_CODE 0x03`

7.663.1.12 `#define PRESET_DICT 0x20`

7.663.1.13 `#define STATIC_TREES 1`

7.663.1.14 `#define STORED_BLOCK 0`

7.663.1.15 `#define Trace( x )`

7.663.1.16 `#define Tracec( c, x )`

7.663.1.17 `#define Tracecv( c, x )`

7.663.1.18 `#define Tracev( x )`

7.663.1.19 `#define Tracevv( x )`

7.663.1.20 `#define TRY_FREE( s, p ) { if (p) ZFREE(s, p); }`

7.663.1.21 `#define ZALLOC( strm, items, size ) (*((strm)->zalloc))((strm)->opaque, (items), (size))`

7.663.1.22 `#define ZFREE( strm, addr ) (*((strm)->zfree))((strm)->opaque, (voidpf)(addr))`

7.663.1.23 `#define ZLIB_INTERNAL`

### 7.663.2 Typedef Documentation

---

Generated on Tue Mar 8 2011 17:10:05 for activemq-cpp-3.2.5 by Doxygen

7.663.2.1 `typedef unsigned char uch`

7.663.2.2 `typedef uch FAR uchf`

7.663.2.3 `typedef unsigned long ulg`

7.663.2.4 `typedef unsigned short ush`

```
#include <vector>
```

## Data Structures

- class **decaf::io::BlockingByteArrayInputStream**

*This is a blocking version of a byte buffer stream.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.665 src/main/decaf/io/BufferedInputStream.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/io/FilterInputStream.h>
```

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

## Data Structures

- class **decaf::io::BufferedInputStream**

*A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.666 src/main/decaf/io/BufferedOutputStream.h File Reference

```
#include <decaf/io/FilterOutputStream.h>
```

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

## Data Structures

- class **decaf::io::BufferedOutputStream**

*Wrapper around another output stream that buffers output before writing to the target output stream.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.667 src/main/decaf/io/ByteArrayInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/util/concurrent/Mutex.h>
#include <vector>
```

## Data Structures

- class **decaf::io::ByteArrayInputStream**

*A **ByteArrayInputStream** (p. 1038) contains an internal buffer that contains bytes that may be read from the stream.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.668 src/main/decaf/io/ByteArrayOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <utility>
```

## Data Structures

- class **decaf::io::ByteArrayOutputStream**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**

## 7.669 src/main/decaf/io/DataInput.h File Reference

```
#include <decaf/util/Config.h>
#include <vector>
#include <string>
#include <decaf/io/IOException.h>
#include <decaf/io/EOFException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

## Data Structures

- class **decaf::io::DataInput**  
*The **DataInput** (p. 1603) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**

## 7.670 src/main/decaf/io/DataInputStream.h File Reference

```
#include <decaf/io/FilterInputStream.h>
```

```
#include <decaf/io/IOException.h>
#include <decaf/io/EOFException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

## Data Structures

- class **decaf::io::DataInputStream**

*A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.671 src/main/decaf/io/DataOutput.h File Reference

```
#include <decaf/util/Config.h>
#include <vector>
#include <string>
#include <decaf/io/IOException.h>
#include <decaf/io/EOFException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

## Data Structures

- class **decaf::io::DataOutput**

*The **DataOutput** (p. 1622) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.672 src/main/decaf/io/DataOutputStream.h File Reference

```
#include <decaf/io/FilterOutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <string>
```

## Data Structures

- class **decaf::io::DataOutputStream**

*A data output stream lets an application write primitive Java data types to an output stream in a portable way.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.673 src/main/decaf/io/EOFException.h File Reference

```
#include <decaf/io/IOException.h>
```

## Data Structures

- class **decaf::io::EOFException**

## Namespaces

- namespace **decaf**



*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.674 src/main/decaf/io/FileDescriptor.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::io::FileDescriptor**

*This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.675 src/main/decaf/io/FilterInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **decaf::io::FilterInputStream**

*A **FilterInputStream** (p. 1950) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.676 src/main/decaf/io/FilterOutputStream.h File Reference

```
#include <decaf/io/OutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **decaf::io::FilterOutputStream**

*This class is the superclass of all classes that filter output streams.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.677 src/main/decaf/io/Flushable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::io::Flushable**

*A **Flushable** (p. 1998) is a destination of data that can be flushed.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.678 src/main/decaf/io/InputStream.h File Reference

```
#include <decaf/io/IOException.h>
#include <decaf/io/Closeable.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::io::InputStream**

*A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.679 src/main/decaf/io/InputStreamReader.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/Reader.h>
```

### Data Structures

- class **decaf::io::InputStreamReader**

*An **InputStreamReader** (p. 2116) is a bridge from byte streams to character streams.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.680 src/main/decaf/io/InterruptedIOException.h File Reference

```
#include <decaf/lang/Exception.h>
```

```
#include <decaf/io/IOException.h>
```

## Data Structures

- class **decaf::io::InterruptedIOException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.681 src/main/decaf/io/IOException.h File Reference

```
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::io::IOException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.682 src/main/decaf/io/OutputStream.h File Reference

```
#include <decaf/io/Closeable.h>
#include <decaf/io/Flushable.h>
#include <decaf/io/IOException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **decaf::io::OutputStream**

*Base interface for any class that wants to represent an output stream of bytes.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.683 src/main/decaf/io/OutputStreamWriter.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/Writer.h>
```

### Data Structures

- class **decaf::io::OutputStreamWriter**

*A class for turning a character stream into a byte stream.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.684 src/main/decaf/io/PushbackInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/FilterInputStream.h>
```

### Data Structures

- class **decaf::io::PushbackInputStream**

*A **PushbackInputStream** (p. 3231) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.685 src/main/decaf/io/Reader.h File Reference

```
#include <string>
#include <decaf/lang/Readable.h>
#include <decaf/io/Closeable.h>
#include <decaf/io/IOException.h>
#include <decaf/io/InputStream.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

### Data Structures

- class **decaf::io::Reader**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**

## 7.686 src/main/decaf/io/UnsupportedEncodingException.h File Reference

```
#include <decaf/io/IOException.h>
```

## Data Structures

- class **decaf::io::UnsupportedEncodingException**  
*Thrown when the the Character Encoding is not supported.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**

## 7.687 src/main/decaf/io/UTFDataFormatException.h File Reference

```
#include <decaf/io/IOException.h>
```

## Data Structures

- class **decaf::io::UTFDataFormatException**  
*Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.688 src/main/decaf/io/Writer.h File Reference

```
#include <string>
#include <vector>
#include <decaf/io/IOException.h>
#include <decaf/io/Closeable.h>
#include <decaf/io/Flushable.h>
#include <decaf/lang/Appendable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **decaf::io::Writer**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.689 src/main/decaf/lang/Appendable.h File Reference

```
#include <decaf/lang/Exception.h>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::lang::Appendable**

*An object to which char sequences and values can be appended.*



## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.690 src/main/decaf/lang/ArrayPointer.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/System.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/concurrent/atomic/AtomicRefCounter.h>
#include <decaf/util/Comparator.h>
#include <memory>
#include <typeinfo>
#include <algorithm>
```

## Data Structures

- class **decaf::lang::ArrayPointer< T, REFCOUNTER >**

*Decaf's implementation of a Smart **Pointer** (p. 3032) that is a template on a Type and is **Thread** (p. 3876) Safe if the default Reference Counter is used.*

- struct **decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayData**

- class **decaf::lang::ArrayPointerComparator< T, R >**

*This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this **ArrayPointer** (p. 736).*

- struct **std::less< decaf::lang::ArrayPointer< T > >**

*An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**
- namespace **std**

## Functions

- template<typename T, typename R, typename U >  
bool **decaf::lang::operator==** (const ArrayPointer< T, R > &left, const U \*right)
- template<typename T, typename R, typename U >  
bool **decaf::lang::operator==** (const U \*left, const ArrayPointer< T, R > &right)
- template<typename T, typename R, typename U >  
bool **decaf::lang::operator!=** (const ArrayPointer< T, R > &left, const U \*right)
- template<typename T, typename R, typename U >  
bool **decaf::lang::operator!=** (const U \*left, const ArrayPointer< T, R > &right)

## 7.691 src/main/decaf/lang/Boolean.h File Reference

```
#include <string>
#include <decaf/lang/Comparable.h>
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::lang::Boolean**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.692 src/main/decaf/lang/Byte.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

## Data Structures

- class **decaf::lang::Byte**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.693 src/main/decaf/lang/Character.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <string>
```

## Data Structures

- class **decaf::lang::Character**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.694 src/main/decaf/lang/CharSequence.h File Reference

```
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::lang::CharSequence**  
*A **CharSequence** (p. 1167) is a readable sequence of char values.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.695 src/main/decaf/lang/Comparable.h File Reference

```
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::lang::Comparable**< **T** >

*This interface imposes a total ordering on the objects of each class that implements it.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.696 src/main/decaf/lang/Double.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/lang/Comparable.h>
```

```
#include <decaf/lang/Number.h>
```

```
#include <decaf/lang/exceptions/NumberFormatException.h>
```

```
#include <string>
```

## Data Structures

- class **decaf::lang::Double**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.697 src/main/decaf/lang/Exception.h File Reference

```
#include <decaf/lang/Throwable.h>
#include <decaf/lang/exceptions/ExceptionDefines.h>
#include <decaf/util/Config.h>
#include <stdarg.h>
#include <sstream>
```

## Data Structures

- class **decaf::lang::Exception**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.698 src/main/decaf/lang/exceptions/ClassCastException.h File Reference

```
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::lang::exceptions::ClassCastException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.699 src/main/decaf/lang/exceptions/IllegalArgumentException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::IllegalArgumentException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.700 src/main/decaf/lang/exceptions/IllegalMonitorStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::IllegalMonitorStateException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.701 src/main/decaf/lang/exceptions/IllegalThreadStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::IllegalThreadStateException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.702 src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::IndexOutOfBoundsException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.703 src/main/decaf/lang/exceptions/InterruptedException.h File Reference

```
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::lang::exceptions::InterruptedException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

### 7.704 src/main/decaf/lang/exceptions/InvalidStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::lang::exceptions::InvalidStateException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

### 7.705 src/main/decaf/lang/exceptions/NoSuchElementException.h File Reference

```
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::lang::exceptions::NoSuchElementException**



## **7.706 src/main/decaf/lang/exceptions/NullPointerException.h File Reference 4665**

### **Namespaces**

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## **7.706 src/main/decaf/lang/exceptions/NullPointerException.h File Reference**

```
#include <decaf/lang/Exception.h>
```

### **Data Structures**

- class **decaf::lang::exceptions::NullPointerException**

### **Namespaces**

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## **7.707 src/main/decaf/lang/exceptions/NumberFormatException.h File Reference**

```
#include <decaf/lang/Exception.h>
```

### **Data Structures**

- class **decaf::lang::exceptions::NumberFormatException**

### **Namespaces**

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.708 src/main/decaf/lang/exceptions/RuntimeException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::RuntimeException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.709 src/main/decaf/lang/Float.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/lang/Number.h>  
#include <decaf/lang/Comparable.h>  
#include <decaf/lang/exceptions/NumberFormatException.h>  
#include <string>
```

### Data Structures

- class **decaf::lang::Float**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.710 src/main/decaf/lang/Integer.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <string>
#include <decaf/lang/exceptions/NumberFormatException.h>
```

### Data Structures

- class **decaf::lang::Integer**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.711 src/main/decaf/lang/Iterable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
```

### Data Structures

- class **decaf::lang::Iterable< E >**  
*Implementing this interface allows an object to be cast to an **Iterable** (p. 2220) type for generic collections API calls.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.712 src/main/decaf/lang/Long.h File Reference

```
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

### Data Structures

- class **decaf::lang::Long**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.713 src/main/decaf/lang/Math.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::lang::Math**  
*The class **Math** (p. 2575) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.714 src/main/decaf/lang/Number.h File Reference

```
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::lang::Number**

*The abstract class **Number** (p. 2918) is the superclass of classes **Byte** (p. 969), **Double** (p. 1841), **Float** (p. 1961), **Integer** (p. 2143), **Long** (p. 2495), and **Short** (p. 3538).*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.715 src/main/decaf/lang/Pointer.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/ClassCastException.h>
#include <decaf/util/concurrent/atomic/AtomicRefCounter.h>
#include <decaf/util/Comparator.h>
#include <memory>
#include <typeinfo>
#include <algorithm>
```

## Data Structures

- struct **decaf::lang::STATIC\_CAST\_TOKEN**
- struct **decaf::lang::DYNAMIC\_CAST\_TOKEN**
- class **decaf::lang::Pointer< T, REFCOUNTER >**

*Decaf's implementation of a Smart **Pointer** (p. 3032) that is a template on a Type and is **Thread** (p. 3876) Safe if the default Reference Counter is used.*

- class **decaf::lang::PointerComparator< T, R >**

*This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 3032) instance.*

- struct **std::less< decaf::lang::Pointer< T > >**

*An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**
- namespace **std**

## Functions

- template<typename T, typename R, typename U >  
bool **decaf::lang::operator==** (const Pointer< T, R > &left, const U \*right)
- template<typename T, typename R, typename U >  
bool **decaf::lang::operator==** (const U \*left, const Pointer< T, R > &right)
- template<typename T, typename R, typename U >  
bool **decaf::lang::operator!=** (const Pointer< T, R > &left, const U \*right)
- template<typename T, typename R, typename U >  
bool **decaf::lang::operator!=** (const U \*left, const Pointer< T, R > &right)

## 7.716 src/main/decaf/lang/Readable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

## Data Structures

- class **decaf::lang::Readable**

*A **Readable** (p. 3251) is a source of characters.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**
- namespace **decaf::lang**

## 7.717 src/main/decaf/lang/Runnable.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::lang::Runnable**  
*Interface for a runnable object - defines a task that can be run by a thread.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.718 src/main/decaf/lang/Runtime.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::lang::Runtime**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.719 src/main/decaf/lang/Short.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

## Data Structures

- class **decaf::lang::Short**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.720 src/main/decaf/lang/String.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/CharSequence.h>
#include <decaf/lang/Comparable.h>
#include <string>
```

## Data Structures

- class **decaf::lang::String**  
*The **String** (p. 3775) class represents an immutable sequence of chars.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.721 src/main/decaf/lang/System.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Map.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```



```
#include <decaf/internal/AprPool.h>
#include <string>
```

## Data Structures

- class **decaf::lang::System**

The **System** (p. 3838) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.

## Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

## 7.722 src/main/decaf/lang/Thread.h File Reference

```
#include <decaf/lang/exceptions/IllegalThreadStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::lang::Thread**

A **Thread** (p. 3876) is a concurrent unit of execution.

- class **decaf::lang::Thread::UncaughtExceptionHandler**

Interface for handlers invoked when a **Thread** (p. 3876) abruptly terminates due to an uncaught exception.

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**
- namespace **decaf::lang**

## 7.723 src/main/decaf/lang/ThreadGroup.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::lang::ThreadGroup**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.724 src/main/decaf/lang/Throwable.h File Reference

```
#include <string>
#include <vector>
#include <iostream>
#include <exception>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::lang::Throwable**

*This class represents an error that has occurred.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.725 src/main/decaf/net/BindException.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/net/SocketException.h>
```

## Data Structures

- class **decaf::net::BindException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.726 src/main/decaf/net/ConnectException.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/net/SocketException.h>
```

## Data Structures

- class **decaf::net::ConnectException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.727 src/main/decaf/net/HttpRetryException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::net::HttpRetryException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.728 src/main/decaf/net/Inet4Address.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/InetAddress.h>
```

### Data Structures

- class **decaf::net::Inet4Address**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.729 src/main/decaf/net/Inet6Address.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/InetAddress.h>
```

### Data Structures

- class **decaf::net::Inet6Address**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.730 src/main/decaf/net/InetAddress.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/lang/ArrayPointer.h>
```

## Data Structures

- class **decaf::net::InetAddress**  
*Represents an IP address.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.731 src/main/decaf/net/InetSocketAddress.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/net/SocketAddress.h>  
#include <string>
```

## Data Structures

- class **decaf::net::InetSocketAddress**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

### 7.732 src/main/decaf/net/MalformedURLException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

#### Data Structures

- class **decaf::net::MalformedURLException**

#### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

### 7.733 src/main/decaf/net/NoRouteToHostException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

#### Data Structures

- class **decaf::net::NoRouteToHostException**

#### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

### 7.734 src/main/decaf/net/PortUnreachableException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

## Data Structures

- class **decaf::net::PortUnreachableException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.735 src/main/decaf/net/ProtocolException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

## Data Structures

- class **decaf::net::ProtocolException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.736 src/main/decaf/net/ServerSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/InetAddress.h>
#include <decaf/net/SocketImpl.h>
#include <decaf/net/SocketImplFactory.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/net/UnknownHostException.h>
#include <decaf/net/SocketTimeoutException.h>
#include <decaf/io/IOException.h>
```

```
#include <string>
```

## Data Structures

- class **decaf::net::ServerSocket**  
*This class implements server sockets.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.737 src/main/decaf/net/ServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/net/InetAddress.h>
```

## Data Structures

- class **decaf::net::ServerSocketFactory**  
*Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.738 src/main/decaf/net/Socket.h File Reference

```
#include <decaf/net/InetAddress.h>  
#include <decaf/net/SocketImplFactory.h>  
#include <decaf/net/SocketException.h>
```



```
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/Closeable.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/net/UnknownHostException.h>
#include <decaf/net/SocketTimeoutException.h>
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::net::Socket**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.739 src/main/decaf/net/SocketAddress.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::net::SocketAddress**  
*Base class for protocol specific **Socket** (p. 3607) addresses.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.740 src/main/decaf/net/SocketError.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::net::SocketError**  
*Static utility class to simplify handling of error codes for socket operations.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.741 src/main/decaf/net/SocketException.h File Reference

```
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::net::SocketException**  
*Exception for errors when manipulating sockets.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.742 src/main/decaf/net/SocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/net/UnknownHostException.h>
```

## Data Structures

- class **decaf::net::SocketFactory**

*The **SocketFactory** (p.3629) is used to create **Socket** (p.3607) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

## 7.743 src/main/decaf/net/SocketImpl.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/FileDescriptor.h>
#include <decaf/net/SocketException.h>
#include <decaf/net/SocketTimeoutException.h>
#include <decaf/net/SocketOptions.h>
#include <string>
```

## Data Structures

- class **decaf::net::SocketImpl**

*Acts as a base class for all physical **Socket** (p.3607) implementations.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

## 7.744 src/main/decaf/net/SocketImplFactory.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::net::SocketImplFactory**  
*Factory class interface for a Factory that creates SocketImpl objects.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.745 src/main/decaf/net/SocketOptions.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::net::SocketOptions**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.746 src/main/decaf/net/SocketTimeoutException.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/io/InterruptedIOException.h>
```

### Data Structures

- class **decaf::net::SocketTimeoutException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

## 7.747 src/main/decaf/net/ssl/SSLContext.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLContextSpi.h>
```

## Data Structures

- class **decaf::net::ssl::SSLContext**

*Represents on implementation of the Secure **Socket** (p.3607) Layer for streaming based sockets.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**
- namespace **decaf::net::ssl**

## 7.748 src/main/decaf/net/ssl/SSLContextSpi.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/SecureRandom.h>
```

## Data Structures

- class **decaf::net::ssl::SSLContextSpi**

*Defines the interface that should be provided by an **SSLContext** (p.3653) provider.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**
- namespace **decaf::net::ssl**

## 7.749 src/main/decaf/net/ssl/SSLParameters.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
#include <vector>
```

## Data Structures

- class **decaf::net::ssl::SSLParameters**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**
- namespace **decaf::net::ssl**

## 7.750 src/main/decaf/net/ssl/SSLServerSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ServerSocket.h>
```

## Data Structures

- class **decaf::net::ssl::SSLServerSocket**

*Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**
- namespace **decaf::net::ssl**

## 7.751 src/main/decaf/net/ssl/SSLServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ServerSocketFactory.h>
#include <vector>
#include <string>
```

## Data Structures

- class **decaf::net::ssl::SSLServerSocketFactory**  
*Factory class interface that provides methods to create SSL Server Sockets.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**
- namespace **decaf::net::ssl**

## 7.752 src/main/decaf/net/ssl/SSLSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/Socket.h>
#include <decaf/net/ssl/SSLParameters.h>
```

## Data Structures

- class **decaf::net::ssl::SSLSocket**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**
- namespace **decaf::net::ssl**

## 7.753 src/main/decaf/net/ssl/SSLSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketFactory.h>
#include <vector>
#include <string>
```

## Data Structures

- class **decaf::net::ssl::SSLSocketFactory**

*Factory class interface for a **SocketFactory** (p. 3629) that can create **SSLSocket** (p. 3670) objects.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**
- namespace **decaf::net::ssl**

## 7.754 src/main/decaf/net/UnknownHostException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

## Data Structures

- class **decaf::net::UnknownHostException**



## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.755 src/main/decaf/net/UnknownServiceException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

## Data Structures

- class **decaf::net::UnknownServiceException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.756 src/main/decaf/net/URI.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/net/URISyntaxException.h>
#include <decaf/net/MalformedURLException.h>
#include <decaf/net/URL.h>
#include <decaf/internal/net/URIType.h>
#include <string>
```

## Data Structures

- class **decaf::net::URI**  
*This class represents an instance of a **URI** (p. 4031) as defined by RFC 2396.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

## 7.757 src/main/decaf/net/URISyntaxException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::net::URISyntaxException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

## 7.758 src/main/decaf/net/URL.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
```

## Data Structures

- class **decaf::net::URL**

*Class **URL** (p. 4072) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

## 7.759 src/main/decaf/net/URLDecoder.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
```

### Data Structures

- class **decaf::net::URLDecoder**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.760 src/main/decaf/net/URLEncoder.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
```

### Data Structures

- class **decaf::net::URLEncoder**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.761 src/main/decaf/nio/Buffer.h File Reference

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/nio/InvalidMarkException.h>
```

## Data Structures

- class **decaf::nio::Buffer**

*A container for data of a specific primitive type.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.762 src/main/decaf/nio/BufferOverflowException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::nio::BufferOverflowException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.763 src/main/decaf/nio/BufferUnderflowException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::nio::BufferUnderflowException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.764 src/main/decaf/nio/ByteBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

## Data Structures

- class **decaf::nio::ByteBuffer**

*This class defines six categories of operations upon byte buffers:*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.765 src/main/decaf/nio/CharBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
```

```
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/lang/CharSequence.h>
#include <decaf/lang/Appendable.h>
```

## Data Structures

- class **decaf::nio::CharBuffer**

*This class defines four categories of operations upon character buffers:*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.766 src/main/decaf/nio/DoubleBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

## Data Structures

- class **decaf::nio::DoubleBuffer**

*This class defines four categories of operations upon double buffers:*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.767 src/main/decaf/nio/FloatBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

### Data Structures

- class **decaf::nio::FloatBuffer**

*This class defines four categories of operations upon float buffers:*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.768 src/main/decaf/nio/IntBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

## Data Structures

- class **decaf::nio::IntBuffer**

*This class defines four categories of operations upon int buffers:*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.769 src/main/decaf/nio/InvalidMarkException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

## Data Structures

- class **decaf::nio::InvalidMarkException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.770 src/main/decaf/nio/LongBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```



## Data Structures

- class **decaf::nio::LongBuffer**

*This class defines four categories of operations upon long long buffers:*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.771 src/main/decaf/nio/ReadOnlyBufferException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

## Data Structures

- class **decaf::nio::ReadOnlyBufferException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.772 src/main/decaf/nio/ShortBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

## Data Structures

- class **decaf::nio::ShortBuffer**

*This class defines four categories of operations upon short buffers:*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.773 src/main/decaf/security/auth/x500/X500Principal.h File Reference

```
#include <string>
#include <vector>
#include <decaf/security/Principal.h>
#include <decaf/util/Map.h>
#include <decaf/io/InputStream.h>
```

## Data Structures

- class **decaf::security::auth::x500::X500Principal**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::security**
- namespace **decaf::security::auth**
- namespace **decaf::security::auth::x500**

## 7.774 src/main/decaf/security/cert/Certificate.h File Reference

```
#include <vector>
#include <decaf/util/Config.h>
```

```
#include <decaf/security/InvalidKeyException.h>
#include <decaf/security/NoSuchAlgorithmException.h>
#include <decaf/security/SignatureException.h>
#include <decaf/security/cert/CertificateEncodingException.h>
#include <decaf/security/cert/CertificateException.h>
```

## Data Structures

- class **decaf::security::cert::Certificate**  
*Base interface for all identity certificates.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

## 7.775 src/main/decaf/security/cert/CertificateEncodingException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/cert/CertificateException.h>
```

## Data Structures

- class **decaf::security::cert::CertificateEncodingException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

## 7.776 src/main/decaf/security/cert/CertificateException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

### Data Structures

- class **decaf::security::cert::CertificateException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

## 7.777 src/main/decaf/security/cert/CertificateExpiredException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/cert/CertificateException.h>
```

### Data Structures

- class **decaf::security::cert::CertificateExpiredException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

## 7.778 src/main/decaf/security/cert/CertificateNotYetValidException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/cert/CertificateException.h>
```

### Data Structures

- class **decaf::security::cert::CertificateNotYetValidException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

## 7.779 src/main/decaf/security/cert/CertificateParsingException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/cert/CertificateException.h>
```

### Data Structures

- class **decaf::security::cert::CertificateParsingException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

## 7.780 src/main/decaf/security/cert/X509Certificate.h File Reference

```
#include <decaf/security/cert/Certificate.h>
#include <decaf/util/Config.h>
#include <decaf/util/Date.h>
```

### Data Structures

- class **decaf::security::cert::X509Certificate**  
*Base interface for all identity certificates.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

## 7.781 src/main/decaf/security/GeneralSecurityException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::security::GeneralSecurityException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.782 src/main/decaf/security/InvalidKeyException.h File Reference

```
#include <decaf/security/KeyException.h>
```

## Data Structures

- class **decaf::security::InvalidKeyException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.783 src/main/decaf/security/Key.h File Reference

```
#include <vector>
#include <string>
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::security::Key**  
*The **Key** (p. 2364) interface is the top-level interface for all keys.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.784 src/main/decaf/security/KeyException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

## Data Structures

- class **decaf::security::KeyException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

### 7.785 src/main/decaf/security/KeyManagementException.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/security/KeyException.h>
```

## Data Structures

- class **decaf::security::KeyManagementException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

### 7.786 src/main/decaf/security/NoSuchAlgorithmException.h File Reference

```
#include <decaf/security/GeneralSecurityException.h>
```

## Data Structures

- class **decaf::security::NoSuchAlgorithmException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**



## **7.787 src/main/decaf/security/NoSuchProviderException.h File Reference**

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

### **Data Structures**

- class **decaf::security::NoSuchProviderException**

### **Namespaces**

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## **7.788 src/main/decaf/security/Principal.h File Reference**

```
#include <decaf/util/Config.h>
```

### **Data Structures**

- class **decaf::security::Principal**  
*Base interface for a principal, which can represent an individual or organization.*

### **Namespaces**

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## **7.789 src/main/decaf/security/PublicKey.h File Reference**

```
#include <decaf/security/Key.h>
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::security::PublicKey**  
*A public key.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.790 src/main/decaf/security/SecureRandom.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Random.h>
#include <decaf/security/SecureRandomSpi.h>
#include <memory>
```

## Data Structures

- class **decaf::security::SecureRandom**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.791 src/main/decaf/security/SecureRandomSpi.h File Reference

```
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::security::SecureRandomSpi**  
*Interface class used by Security Service Providers to implement a source of secure random bytes.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::security**

## 7.792 src/main/decaf/security/SignatureException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

## Data Structures

- class **decaf::security::SignatureException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::security**

## 7.793 src/main/decaf/util/AbstractCollection.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Collection.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <memory>
```

## Data Structures

- class **decaf::util::AbstractCollection**< E >

*This class provides a skeletal implementation of the **Collection** (p. 1216) interface, to minimize the effort required to implement this interface.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.794 src/main/decaf/util/AbstractList.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/List.h>
#include <memory>
```

## Data Structures

- class **decaf::util::AbstractList**< E >

*This class provides a skeletal implementation of the **List** (p. 2409) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.795 src/main/decaf/util/AbstractMap.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Map.h>
#include <decaf/util/Set.h>
#include <memory>
```

### Data Structures

- class **decaf::util::AbstractMap**< **K**, **V**, **COMPARATOR** >

*This class provides a skeletal implementation of the **Map** (p. 2538) interface, to minimize the effort required to implement this interface.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.796 src/main/decaf/util/AbstractQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Queue.h>
#include <memory>
```

## Data Structures

- class **decaf::util::AbstractQueue**< **E** >

*This class provides skeletal implementations of some **Queue** (p. 3239) operations.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.797 src/main/decaf/util/AbstractSequentialList.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractList.h>
#include <memory>
```

## Data Structures

- class **decaf::util::AbstractSequentialList**< **E** >

*This class provides a skeletal implementation of the **List** (p. 2409) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.798 src/main/decaf/util/AbstractSet.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Set.h>
#include <memory>
```

### Data Structures

- class **decaf::util::AbstractSet**< E >

*This class provides a skeletal implementation of the **Set** (p. 3538) interface to minimize the effort required to implement this interface.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.799 src/main/decaf/util/Collection.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/concurrent/Synchronizable.h>
```

### Data Structures

- class **decaf::util::Collection**< E >

*The root interface in the collection hierarchy.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.800 src/main/decaf/util/Comparator.h File Reference

```
#include <decaf/util/Config.h>
#include <algorithm>
#include <functional>
```

## Data Structures

- class **decaf::util::Comparator**< T >

*A comparison function, which imposes a total ordering on some collection of objects.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.801 src/main/decaf/util/comparators/Less.h File Reference

```
#include <decaf/util/Comparator.h>
```

## Data Structures

- class **decaf::util::comparators::Less**< E >

*Simple **Less** (p. 2399) **Comparator** (p. 1251) that compares to elements to determine if the first is less than the second.*



## 7.802 src/main/decaf/util/concurrent/atomic/AtomicBoolean.h File Reference 4713

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::comparators**

## 7.802 src/main/decaf/util/concurrent/atomic/AtomicBoolean.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::atomic::AtomicBoolean**  
*A boolean value that may be updated atomically.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

## 7.803 src/main/decaf/util/concurrent/atomic/AtomicInteger.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <string>
```

### Data Structures

- class **decaf::util::concurrent::atomic::AtomicInteger**

*An int value that may be updated atomically.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

## 7.804 src/main/decaf/util/concurrent/atomic/AtomicRefCounter.h File Reference

```
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
```

## Data Structures

- class **decaf::util::concurrent::atomic::AtomicRefCounter**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

## 7.805 src/main/decaf/util/concurrent/atomic/AtomicReference.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/lang/Long.h>
```

```
#include <apr_atomic.h>
```

## Data Structures

- class **decaf::util::concurrent::atomic::AtomicReference**< T >

*An Pointer reference that may be updated atomically.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

## 7.806 src/main/decaf/util/concurrent/BlockingQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/AbstractQueue.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/lang/exceptions/InterruptedException.h>
```

## Data Structures

- class **decaf::util::concurrent::BlockingQueue**< E >

*A **decaf::util::Queue** (p. 3239) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.807 src/main/decaf/util/concurrent/BrokenBarrierException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::util::concurrent::BrokenBarrierException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.808 src/main/decaf/util/concurrent/Callable.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::util::concurrent::Callable< V >**  
*A task that returns a result and may throw an exception.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.809 src/main/decaf/util/concurrent/CancellationException.h File Reference

```
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::util::concurrent::CancellationException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.810 src/main/decaf/util/concurrent/Concurrent.h File Reference

```
#include <decaf/util/concurrent/Lock.h>
```

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## Defines

- #define **WAIT\_INFINITE** 0xFFFFFFFF  
*The synchronized macro defines a mechanism for synchronizing a section of code.*
- #define **synchronized(W)**

### 7.810.1 Define Documentation

#### 7.810.1.1 #define synchronized( W )

##### Value:

```
if(false){} \
else \
for( decaf::util::concurrent::Lock lock_W(W); \
    lock_W.isLocked(); lock_W.unlock() )
```

### 7.810.1.2 #define WAIT\_INFINITE 0xFFFFFFFF

The synchronized macro defines a mechanism for synchronizing a section of code.

The macro must be passed an object that implements the Synchronizable interface.

The macro works by creating a for loop that will loop exactly once, creating a Lock object that is scoped to the loop. Once the loop completes and exits the Lock object goes out of scope releasing the lock on object W. For added safety the if else is used because not all compilers restrict the lifetime of loop variables to the loop, they will however restrict them to the scope of the else.

The macro would be used as follows.

Synchronizable X;

```
somefunction() { synchronized(X) (p. 4713) { // Do something that needs synchronizing. } }
```

## 7.811 src/main/decaf/util/concurrent/ConcurrentMap.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Map.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

### Data Structures

- class **decaf::util::concurrent::ConcurrentMap**< K, V, COMPARATOR >

*Interface for a **Map** (p. 2538) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 2538) interface.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.812 src/main/decaf/util/concurrent/ConcurrentStlMap.h File Reference

```
#include <map>
#include <vector>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/ConcurrentMap.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Map.h>
```

### Data Structures

- class **decaf::util::concurrent::ConcurrentStlMap**< **K**, **V**, **COMPARATOR** >  
*Map* (p. 2538) *template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.813 src/main/decaf/util/concurrent/CountDownLatch.h File Reference

```
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::util::concurrent::CountDownLatch**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.814 src/main/decaf/util/concurrent/Delayed.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/util/concurrent/TimeUnit.h>
```

## Data Structures

- class **decaf::util::concurrent::Delayed**

*A mix-in style interface for marking objects that should be acted upon after a given delay.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.815 src/main/decaf/util/concurrent/ExecutionException.h File Reference

```
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::util::concurrent::ExecutionException**



## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.816 src/main/decaf/util/concurrent/Executor.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/util/concurrent/RejectedExecutionException.h>
```

## Data Structures

- class **decaf::util::concurrent::Executor**

*An object that executes submitted **decaf.lang.Runnable** (p. 3418) tasks.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.817 src/main/decaf/util/concurrent/ExecutorService.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/Executor.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/lang/exceptions/InterruptedException.h>
```

## Data Structures

- class **decaf::util::concurrent::ExecutorService**

An *Executor* (p. 1926) that provides methods to manage termination and methods that can produce a *Future* (p. 2029) for tracking progress of one or more asynchronous tasks.

## Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.818 src/main/decaf/util/concurrent/Future.h File Reference

### Data Structures

- class **decaf::util::concurrent::Future< V >**

A *Future* (p. 2029) represents the result of an asynchronous computation.

### Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.819 src/main/decaf/util/concurrent/Lock.h File Reference

```
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::Lock**

*A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.820 src/main/decaf/util/concurrent/locks/Lock.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/util/concurrent/locks/Condition.h>
```

## Data Structures

- class **decaf::util::concurrent::locks::Lock**  
*Lock (p. 2452) implementations provide more extensive locking operations than can be obtained using synchronized statements.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

## 7.821 src/main/decaf/util/concurrent/locks/Condition.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Date.h>
```

```
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/IllegalMonitorStateException.h>
```

## Data Structures

- class **decaf::util::concurrent::locks::Condition**

*Condition* (p. 1282) factors out the *Mutex* (p. 2866) monitor methods (*wait*, *notify* and *notifyAll*) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary *Lock* (p. 2452) implementations.

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

## 7.822 src/main/decaf/util/concurrent/locks/LockSupport.h File Reference

```
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::util::concurrent::locks::LockSupport**

*Basic thread blocking primitives for creating locks and other synchronization classes.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**
- namespace **decaf::util**

## 7.823 src/main/decaf/util/concurrent/locks/ReadWriteLock.h File Reference 4725

- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

## 7.823 src/main/decaf/util/concurrent/locks/ReadWriteLock.h File Reference

### Data Structures

- class **decaf::util::concurrent::locks::ReadWriteLock**  
*A **ReadWriteLock** (p. 3263) maintains a pair of associated locks, one for read-only operations and one for writing.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

## 7.824 src/main/decaf/util/concurrent/locks/ReentrantLock.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/locks/Lock.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **decaf::util::concurrent::locks::ReentrantLock**  
*A reentrant mutual exclusion **Lock** (p. 2452) with extended capabilities.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

## 7.825 src/main/decaf/util/concurrent/Mutex.h File Reference

```
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/lang/Thread.h>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::Mutex**

***Mutex** (p. 2866) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.826 src/main/decaf/util/concurrent/PooledThread.h File Reference

```
#include <decaf/lang/Thread.h>
#include <decaf/util/concurrent/PooledThreadListener.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::util::concurrent::PooledThread**

## 7.827 src/main/decaf/util/concurrent/PooledThreadListener.h File Reference 4727

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.827 src/main/decaf/util/concurrent/PooledThreadListener.h File Reference

```
#include <decaf/lang/Exception.h>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::PooledThreadListener**  
*Abstract Listener Interface for users of **ThreadPool** (p. 3888).*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.828 src/main/decaf/util/concurrent/RejectedExecutionException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::util::concurrent::RejectedExecutionException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.829 src/main/decaf/util/concurrent/RejectedExecutionHandler.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/RejectedExecutionException.h>
```

## Data Structures

- class **decaf::util::concurrent::RejectedExecutionHandler**

*A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. ??).*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.830 src/main/decaf/util/concurrent/Semaphore.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <memory>
```



## Data Structures

- class **decaf::util::concurrent::Semaphore**

*A counting semaphore.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.831 src/main/decaf/util/concurrent/Synchronizable.h File Reference

```
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalMonitorStateException.h>
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::util::concurrent::Synchronizable**

*The interface for all synchronizable objects (that is, objects that can be locked and unlocked).*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.832 src/main/decaf/util/concurrent/SynchronousQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/BlockingQueue.h>
#include <vector>
```

### Data Structures

- class **decaf::util::concurrent::SynchronousQueue< E >**  
*A **blocking queue** (p. 848) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.*
- class **decaf::util::concurrent::SynchronousQueue< E >::EmptyIterator**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.833 src/main/decaf/util/concurrent/TaskListener.h File Reference

```
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Exception.h>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::TaskListener**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.834 src/main/decaf/util/concurrent/ThreadFactory.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::ThreadFactory**

*public interface ThreadFactory* (p. 3887)

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.835 src/main/decaf/util/concurrent/ThreadPool.h File Reference

```
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/PooledThread.h>
#include <decaf/util/concurrent/PooledThreadListener.h>
#include <decaf/util/concurrent/TaskListener.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/util/Config.h>
#include <vector>
```

### Data Structures

- class **decaf::util::concurrent::ThreadPool**

*Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.836 src/main/decaf/util/concurrent/TimeoutException.h File Reference

```
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::util::concurrent::TimeoutException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.837 src/main/decaf/util/concurrent/TimeUnit.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

## Data Structures

- class **decaf::util::concurrent::TimeUnit**

A *TimeUnit* (p. 3919) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.838 src/main/decaf/util/Date.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <string>
```

## Data Structures

- class **decaf::util::Date**  
*Wrapper class around a time value in milliseconds.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.839 src/main/decaf/util/Iterator.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

## Data Structures

- class **decaf::util::Iterator**< **T** >

*Defines an object that can be used to iterate over the elements of a collection.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.840 src/main/decaf/util/List.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractCollection.h>
#include <decaf/util/ListIterator.h>
```

## Data Structures

- class **decaf::util::List**< **E** >

*An ordered collection (also known as a sequence).*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.841 src/main/decaf/util/ListIterator.h File Reference

```
#include <decaf/util/Iterator.h>
```

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

## Data Structures

- class **decaf::util::ListIterator**< E >

*An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.842 src/main/decaf/util/logging/ConsoleHandler.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/logging/StreamHandler.h>
#include <decaf/util/logging/SimpleFormatter.h>
#include <decaf/io/IOException.h>
#include <decaf/internal/io/StandardErrorOutputStream.h>
```

## Data Structures

- class **decaf::util::logging::ConsoleHandler**

*This **Handler** (p. 2042) publishes log records to `System.err`.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.843 src/main/decaf/util/logging/EventManager.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <string>
```

### Data Structures

- class **decaf::util::logging::EventManager**

*ErrorManager* (p. 1884) objects can be attached to *Handlers* to process any error that occur on a *Handler* (p. 2042) during Logging.

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.844 src/main/decaf/util/logging/Filter.h File Reference

```
#include <decaf/util/logging/LogRecord.h>
```

### Data Structures

- class **decaf::util::logging::Filter**

*A Filter* (p. 1949) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::logging**



## 7.845 src/main/decaf/util/logging/Formatter.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/logging/Handler.h>
```

### Data Structures

- class **decaf::util::logging::Formatter**  
*A **Formatter** (p. 2027) provides support for formatting LogRecords.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.846 src/main/decaf/util/logging/Handler.h File Reference

```
#include <decaf/io/Closeable.h>
#include <decaf/lang/Exception.h>
#include <decaf/util/logging/LogRecord.h>
#include <decaf/util/logging/Level.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <string>
```

### Data Structures

- class **decaf::util::logging::Handler**  
*A **Handler** (p. 2042) object takes log messages from a **Logger** (p. 2461) and exports them.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.847 src/main/decaf/util/logging/Level.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

### Data Structures

- class **decaf::util::logging::Level**  
*The **Level** (p. 2403) class defines a set of standard logging levels that can be used to control logging output.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.848 src/main/decaf/util/logging/Logger.h File Reference

```
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/logging/LogRecord.h>
#include <decaf/util/logging/LogManager.h>
#include <decaf/util/logging/Handler.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <list>
#include <string>
#include <stdarg.h>
```

## Data Structures

- class **decaf::util::logging::Logger**

A *Logger* (p.2461) object is used to log messages for a specific system or application component.

## Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.849 src/main/decaf/util/logging/LoggerCommon.h File Reference

### Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

### Enumerations

- enum **decaf::util::logging::Levels** {  
    **decaf::util::logging::Off**, **decaf::util::logging::Null**, **decaf::util::logging::Markblock**,  
    **decaf::util::logging::Debug**,  
    **decaf::util::logging::Info**, **decaf::util::logging::Warn**, **decaf::util::logging::Error**,  
    **decaf::util::logging::Fatal**,  
    **decaf::util::logging::Throwing** }

Defines an enumeration for logging levels.

## 7.850 src/main/decaf/util/logging/LoggerDefines.h File Reference

```
#include <decaf/util/logging/SimpleLogger.h>
#include <sstream>
```

## Defines

- `#define LOGDECAF_DECLARE(loggerName) static decaf::util::logging::SimpleLogger loggerName;`
- `#define LOGDECAF_INITIALIZE(loggerName, className, loggerFamily) decaf::util::logging::SimpleLogger className::loggerName(loggerFamily);`
- `#define LOGDECAF_DECLARE_LOCAL(loggerName) decaf::util::logging::Logger loggerName;`
- `#define LOGDECAF_DEBUG(logger, message) logger.debug(__FILE__, __LINE__, message);`
- `#define LOGDECAF_DEBUG_1(logger, message, value)`
- `#define LOGDECAF_INFO(logger, message) logger.info(__FILE__, __LINE__, message);`
- `#define LOGDECAF_ERROR(logger, message) logger.error(__FILE__, __LINE__, message);`
- `#define LOGDECAF_WARN(logger, message) logger.warn(__FILE__, __LINE__, message);`
- `#define LOGDECAF_FATAL(logger, message) logger.fatal(__FILE__, __LINE__, message);`

### 7.850.1 Define Documentation

7.850.1.1 `#define LOGDECAF_DEBUG( logger, message ) logger.debug(__FILE__, __LINE__, message);`

7.850.1.2 `#define LOGDECAF_DEBUG_1( logger, message, value )`

#### Value:

```

;
{
    std::ostringstream ostream;
    ostream << message << value;
    logger.debug(__FILE__, __LINE__, ostream.str());
}

```

- 7.850.1.3 `#define LOGDECAF_DECLARE( loggerName ) static  
decaf::util::logging::SimpleLogger loggerName;`
- 7.850.1.4 `#define LOGDECAF_DECLARE_LOCAL( loggerName  
) decaf::util::logging::Logger loggerName;`
- 7.850.1.5 `#define LOGDECAF_ERROR( logger, message ) logger.error(__FILE__, __LINE__,  
message);`
- 7.850.1.6 `#define LOGDECAF_FATAL( logger, message ) logger.fatal(__FILE__, __LINE__,  
message);`
- 7.850.1.7 `#define LOGDECAF_INFO( logger, message ) logger.info(__FILE__, __LINE__,  
message);`
- 7.850.1.8 `#define LOGDECAF_INITIALIZE( loggerName, className, loggerFamily  
) decaf::util::logging::SimpleLogger className::loggerName(loggerFamily);`
- 7.850.1.9 `#define LOGDECAF_WARN( logger, message ) logger.warn(__FILE__, __LINE__,  
message);`

## 7.851 src/main/decaf/util/logging/LoggerHierarchy.h File Reference

### Data Structures

- class `decaf::util::logging::LoggerHierarchy`

### Namespaces

- namespace `decaf`  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace `decaf::util`
- namespace `decaf::util::logging`

## 7.852 src/main/decaf/util/logging/LogManager.h File Reference

```
#include <map>
#include <list>
#include <string>
#include <vector>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
```

```
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

## Data Structures

- class **decaf::util::logging::LogManager**

*There is a single global **LogManager** (p.2480) object that is used to maintain a set of shared state about Loggers and log services.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**
- namespace **decaf::io**
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.853 src/main/decaf/util/logging/LogRecord.h File Reference

```
#include <decaf/lang/Throwable.h>
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/logging/Level.h>
#include <decaf/util/Config.h>
#include <memory>
#include <string>
```

## Data Structures

- class **decaf::util::logging::LogRecord**

***LogRecord** (p.2487) objects are used to pass logging requests between the logging framework and individual log Handlers.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.854 src/main/decaf/util/logging/LogWriter.h File Reference

```
#include <decaf/util/concurrent/Mutex.h>
```

## Data Structures

- class **decaf::util::logging::LogWriter**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.855 src/main/decaf/util/logging/MarkBlockLogger.h File Reference

```
#include <decaf/util/logging/Logger.h>
```

## Data Structures

- class **decaf::util::logging::MarkBlockLogger**  
*Defines a class that can be used to mark the entry and exit from scoped blocks.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.856 src/main/decaf/util/logging/PropertiesChangeListener.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::logging::PropertiesChangeListener**

*Defines the interface that classes can use to listen for change events on **Properties** (p. 3216).*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.857 src/main/decaf/util/logging/SimpleFormatter.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/logging/Formatter.h>
#include <string>
```

### Data Structures

- class **decaf::util::logging::SimpleFormatter**

*Print a brief summary of the **LogRecord** (p. 2487) in a human readable format.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::logging**



## 7.858 src/main/decaf/util/logging/SimpleLogger.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::logging::SimpleLogger**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.859 src/main/decaf/util/logging/StreamHandler.h File Reference

```
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/logging/Handler.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::logging::StreamHandler**  
*Stream based logging **Handler** (p. 2042).*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.860 src/main/decaf/util/logging/XMLFormatter.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/logging/Formatter.h>
#include <decaf/util/logging/LogRecord.h>
```

### Data Structures

- class **decaf::util::logging::XMLFormatter**  
*Format a **LogRecord** (p. 2487) into a standard XML format.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.861 src/main/decaf/util/Map.h File Reference

```
#include <functional>
#include <vector>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
```

### Data Structures

- class **decaf::util::Map< K, V, COMPARATOR >**  
***Map** (p. 2538) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*
- class **decaf::util::Map< K, V, COMPARATOR >::Entry**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.862 src/main/decaf/util/PriorityQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Collection.h>
#include <decaf/util/AbstractQueue.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Comparator.h>
#include <decaf/util/comparators/Less.h>
#include <decaf/lang/Math.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <memory>
```

### Data Structures

- class **decaf::util::PriorityQueue< E >**  
*An unbounded priority queue based on a binary heap algorithm.*
- class **decaf::util::PriorityQueue< E >::PriorityQueueIterator**
- class **decaf::util::PriorityQueue< E >::ConstPriorityQueueIterator**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.863 src/main/decaf/util/Properties.h File Reference

```
#include <vector>
#include <string>
```

```
#include <decaf/util/Config.h>
#include <decaf/util/StlMap.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/io/IOException.h>
```

## Data Structures

- class **decaf::util::Properties**

*Java-like properties class for mapping string names to string values.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**
- namespace **decaf::util**

## 7.864 src/main/decaf/util/Random.h File Reference

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/Config.h>
#include <vector>
#include <cmath>
```

## Data Structures

- class **decaf::util::Random**

***Random** (p. 3245) Value Generator which is used to generate a stream of pseudorandom numbers.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.865 src/main/decaf/util/Set.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractCollection.h>
```

## Data Structures

- class **decaf::util::Set< E >**

*A collection that contains no duplicate elements.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.866 src/main/decaf/util/StlList.h File Reference

```
#include <list>
#include <algorithm>
#include <memory>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/concurrent/Synchronizable.h>
```

```
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/ListIterator.h>
#include <decaf/util/List.h>
```

## Data Structures

- class **decaf::util::StlList< E >**  
*List* (p. 2409) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.
- class **decaf::util::StlList< E >::StlListIterator**
- class **decaf::util::StlList< E >::ConstStlListIterator**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.867 src/main/decaf/util/StlMap.h File Reference

```
#include <map>
#include <vector>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Map.h>
```

## Data Structures

- class **decaf::util::StlMap< K, V, COMPARATOR >**  
*Map* (p. 2538) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.868 src/main/decaf/util/StlQueue.h File Reference

```
#include <list>
#include <vector>
#include <decaf/util/Iterator.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::util::StlQueue< T >**  
*The **Queue** (p. 3239) class accepts messages with an **psuh(m)** command where **m** is the message to be queued.*
- class **decaf::util::StlQueue< T >::QueueIterator**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.869 src/main/decaf/util/StlSet.h File Reference

```
#include <set>
#include <vector>
#include <memory>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
```

```
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractSet.h>
```

### Data Structures

- class **decaf::util::StlSet< E >**  
*Set* (p. 3538) *template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set.*
- class **decaf::util::StlSet< E >::SetIterator**
- class **decaf::util::StlSet< E >::ConstSetIterator**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.870 src/main/decaf/util/StringTokenizer.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/Config.h>
#include <string>
```

### Data Structures

- class **decaf::util::StringTokenizer**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.871 src/main/decaf/util/Timer.h File Reference

```
#include <memory>
```



```
#include <decaf/util/Config.h>
#include <decaf/util/Date.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

## Data Structures

- class **decaf::util::Timer**

*A facility for threads to schedule tasks for future execution in a background thread.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.872 src/main/decaf/util/TimerTask.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/Mutex.h>
```

## Data Structures

- class **decaf::util::TimerTask**

*A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 3901).*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**

- namespace **decaf::internal::util**
- namespace **decaf::util**

## 7.873 src/main/decaf/util/UUID.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <apr_uuid.h>
#include <string>
```

### Data Structures

- class **decaf::util::UUID**  
*A class that represents an immutable universally unique identifier (**UUID** (p. 4080)).*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.874 src/main/decaf/util/zip/Adler32.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/zip/Checksum.h>
```

### Data Structures

- class **decaf::util::zip::Adler32**  
*Clas that can be used to compute an Adler-32 **Checksum** (p. 1174) for a data stream.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.875 src/main/decaf/util/zip/CheckedInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/zip/Checksum.h>
#include <decaf/io/FilterInputStream.h>
```

### Data Structures

- class **decaf::util::zip::CheckedInputStream**  
*An implementation of a `FilterInputStream` that will maintain a **Checksum** (p. 1174) of the bytes read, the **Checksum** (p. 1174) can then be used to verify the integrity of the input stream.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.876 src/main/decaf/util/zip/CheckedOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/zip/Checksum.h>
#include <decaf/io/FilterOutputStream.h>
```

### Data Structures

- class **decaf::util::zip::CheckedOutputStream**  
*An implementation of a `FilterOutputStream` that will maintain a **Checksum** (p. 1174) of the bytes written, the **Checksum** (p. 1174) can then be used to verify the integrity of the output stream.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.877 src/main/decaf/util/zip/Checksum.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <vector>
```

## Data Structures

- class **decaf::util::zip::Checksum**

*An interface used to represent **Checksum** (p. 1174) values in the Zip package.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.878 src/main/decaf/util/zip/CRC32.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/zip/Checksum.h>
```

## Data Structures

- class **decaf::util::zip::CRC32**

*Class that can be used to compute a CRC-32 checksum for a data stream.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.879 src/main/decaf/util/zip/DataFormatException.h File Reference

```
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::util::zip::DataFormatException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.880 src/main/decaf/util/zip/Deflater.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <vector>
```

## Data Structures

- class **decaf::util::zip::Deflater**  
*This class compresses data using the DEFLATE algorithm (see specification).*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.881 src/main/decaf/util/zip/DeflaterOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/FilterOutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/util/zip/Deflater.h>
#include <vector>
```

## Data Structures

- class **decaf::util::zip::DeflaterOutputStream**

*Provides a FilterOutputStream instance that compresses the data before writing it to the wrapped OutputStream.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.882 src/main/decaf/util/zip/Inflater.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

```
#include <decaf/util/zip/DataFormatException.h>
#include <vector>
```

## Data Structures

- class **decaf::util::zip::Inflater**

*This class uncompresses data that was compressed using the DEFLATE algorithm (see specification).*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.883 src/main/decaf/util/zip/InflaterInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/FilterInputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/util/zip/Inflater.h>
#include <vector>
```

## Data Structures

- class **decaf::util::zip::InflaterInputStream**

*A FilterInputStream that decompresses data read from the wrapped InputStream instance.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.884 src/main/decaf/util/zip/ZipException.h File Reference

```
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::util::zip::ZipException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::zip**



# Index

~AbstractCollection 257  
     decaf::util::AbstractCollection, 162  
 ~AbstractList 236  
     decaf::util::AbstractList, 174  
 ~AbstractMap 244  
     decaf::util::AbstractMap, 175  
 ~AbstractQueue 249  
     decaf::util::AbstractQueue, 177  
 ~AbstractSequentialList 253  
     decaf::util::AbstractSequentialList, 180  
 ~ActiveMQCPP  
     activemq::library::ActiveMQCPP, 311  
 ~ActiveMQConnection  
     activemq::core::ActiveMQConnection,  
     ~ActiveMQConnectionFactory  
     activemq::core::ActiveMQConnectionFactory,  
     284  
 ~ActiveMQConnectionMetaData  
     activemq::core::ActiveMQConnectionMetaData,  
     ~ActiveMQConsumer  
     activemq::core::ActiveMQConsumer,  
     303  
 ~ActiveMQBlobMessage 186  
     activemq::commands::ActiveMQBlobMessage, 294  
 ~ActiveMQBlobMessageMarshaller 195  
     activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller,  
     203  
     activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller,  
     191  
     activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller,  
     199  
     activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller,  
     208  
     activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller,  
     212  
 ~ActiveMQBytesMessage 333  
     activemq::commands::ActiveMQBytesMessage,  
     219  
 ~ActiveMQBytesMessageMarshaller 345  
     activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller,  
     240  
     activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller,  
     240  
     activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller,  
     236  
     activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller,  
     244  
     activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller,  
     249  
     activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller,  
     253  
     activemq::library::ActiveMQCPP, 311  
     ~ActiveMQConnection  
     activemq::core::ActiveMQConnection,  
     ~ActiveMQConnectionFactory  
     activemq::core::ActiveMQConnectionFactory,  
     284  
     ~ActiveMQConnectionMetaData  
     activemq::core::ActiveMQConnectionMetaData,  
     ~ActiveMQConsumer  
     activemq::core::ActiveMQConsumer,  
     303  
     ~ActiveMQDestination  
     activemq::core::ActiveMQDestination,  
     315  
     ~ActiveMQDestinationMarshaller  
     activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller,  
     329  
     activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller,  
     344  
     activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller,  
     335  
     activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller,  
     337  
     activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller,  
     345  
     ~ActiveMQException  
     activemq::core::ActiveMQException,  
     345

- 349
- ~ActiveMQMapMessage
  - activemq::commands::ActiveMQMapMessage, 463
  - 353
- ~ActiveMQMapMessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller, 476
  - 370
  - ~ActiveMQQueue
  - activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller, 480
  - 383
  - activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller, 480
  - 366
  - activemq::core::ActiveMQQueueBrowser, 484
  - activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller, 484
  - 374
  - ~ActiveMQQueueMarshaller
  - activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller, 492
  - 378
  - activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller, 505
  - 387
- ~ActiveMQMessage
  - activemq::commands::ActiveMQMessage, 488
  - 391
- ~ActiveMQMessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 496
  - 398
  - activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller, 501
  - 411
  - activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller, 509
  - 394
  - activemq::core::ActiveMQSession, 517
  - activemq::wireformat::openwire::marshal::v4::ActiveMQSessionMarshaller, 517
  - 402
  - activemq::core::ActiveMQSessionExecutor, 533
  - activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller, 533
  - 407
  - ~ActiveMQStreamMessage
  - activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller, 539
  - 415
- ~ActiveMQMessageTemplate
  - ~ActiveMQStreamMessageMarshaller
  - activemq::commands::ActiveMQMessageTemplate, 558
  - 422
- ~ActiveMQObjectMessage
  - activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessage, 571
  - 439
  - activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessage, 571
  - 439
- ~ActiveMQObjectMessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller, 554
  - 446
  - activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller, 563
  - 459
  - activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller, 567
  - 442
  - activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller, 575
  - 450
  - activemq::commands::ActiveMQTempDestination, 580
  - activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller, 580
  - 454
  - ~ActiveMQTempDestinationMarshaller

Generated on Tue Mar 8 2011 17:10:05 for activemq-cpp-3.2.5 by Doxygen

- activemq::transport::failover::BackupTransportPool, 760
- ~BackupTransportPool
- activemq::transport::failover::BackupTransportPool, 763
- ~BaseCommand
- activemq::commands::BaseCommand, 766
- ~BaseCommandMarshaller
- activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller, 787
- activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller, 808
- activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller, 772
- activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller, 780
- activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller, 794
- activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller, 801
- ~BaseDataStreamMarshaller
- activemq::wireformat::openwire::marshal::v1::BaseDataStreamMarshaller, 820
- ~BaseDataStructure
- activemq::commands::BaseDataStructure, 838
- ~BindException
- decaf::net::BindException, 844
- ~BlockingByteArrayInputStream
- decaf::io::BlockingByteArrayInputStream, 846
- ~BlockingQueue
- decaf::util::concurrent::BlockingQueue, 852
- ~Boolean
- decaf::lang::Boolean, 857
- ~BooleanExpression
- activemq::commands::BooleanExpression, 862
- ~BooleanStream
- activemq::wireformat::openwire::utils::BooleanStream, 864
- ~BrokenBarrierException
- decaf::util::concurrent::BrokenBarrierException, 868
- ~BrokerError
- activemq::commands::BrokerError, 870
- ~BrokerException
- activemq::exceptions::BrokerException, 874
- ~BrokerId
- activemq::commands::BrokerId, 876
- ~BrokerIdMarshaller
- activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 887
- activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller, 899
- activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller, 879
- activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller, 883
- activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, 891
- activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller, 895
- ~BrokerInfo
- activemq::commands::BrokerInfo, 904
- ~BrokerInfoMarshaller
- activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 920
- ~BrokerInfoStreamMarshaller
- activemq::wireformat::openwire::marshal::v2::BrokerInfoStreamMarshaller, 933
- ~BrokerInfoStructure
- activemq::wireformat::openwire::marshal::v3::BrokerInfoStructure, 912
- ~BrokerInfoStreamStructure
- activemq::wireformat::openwire::marshal::v4::BrokerInfoStreamStructure, 916
- ~BrokerInfoStreamStructure
- activemq::wireformat::openwire::marshal::v5::BrokerInfoStreamStructure, 924
- ~BrokerInfoStreamStructure
- activemq::wireformat::openwire::marshal::v6::BrokerInfoStreamStructure, 929
- ~Buffer
- decaf::nio::Buffer, 939
- ~BufferFactory
- decaf::internal::nio::BufferFactory, 954
- ~BufferOverflowException
- decaf::nio::BufferOverflowException, 966
- ~BufferUnderflowException
- decaf::nio::BufferUnderflowException, 969
- ~BufferedInputStream
- decaf::io::BufferedInputStream, 945
- ~BufferedOutputStream
- decaf::io::BufferedOutputStream, 950
- ~Byte
- decaf::lang::Byte, 972
- ~ByteArrayAdapter

- decaf::internal::util::ByteArrayAdapter, 987
- ~ByteArrayBuffer
  - decaf::internal::nio::ByteArrayBuffer, 1019
- ~ByteArrayInputStream
  - decaf::io::ByteArrayInputStream, 1042
- ~ByteArrayOutputStream
  - decaf::io::ByteArrayOutputStream, 1048
- ~ByteBuffer
  - decaf::nio::ByteBuffer, 1056
- ~BytesMessage
  - cms::BytesMessage, 1083
- ~CMSException
  - cms::CMSException, 1192
- ~CMSExceptionSupport
  - activemq::util::CMSExceptionSupport, 1195
- ~CMSProperties
  - cms::CMSProperties, 1197
- ~CMSSecurityException
  - cms::CMSSecurityException, 1200
- ~CRC32
  - decaf::util::zip::CRC32, 1569
- ~CachedConsumer
  - activemq::cmsutil::CachedConsumer, 1099
- ~CachedProducer
  - activemq::cmsutil::CachedProducer, 1103
- ~Callable
  - decaf::util::concurrent::Callable, 1109
- ~CancellationException
  - decaf::util::concurrent::CancellationException, 1112
- ~Certificate
  - decaf::security::cert::Certificate, 1113
- ~CertificateEncodingException
  - decaf::security::cert::CertificateEncodingException, 1117
- ~CertificateException
  - decaf::security::cert::CertificateException, 1119
- ~CertificateExpiredException
  - decaf::security::cert::CertificateExpiredException, 1122
- ~CertificateNotYetValidException
  - decaf::security::cert::CertificateNotYetValidException, 1124
- ~CertificateParsingException
  - decaf::security::cert::CertificateParsingException, 1126
- ~CharArrayBuffer
  - decaf::internal::nio::CharArrayBuffer, 1141
- ~CharBuffer
  - decaf::nio::CharBuffer, 1152
- ~CharSequence
  - decaf::lang::CharSequence, 1168
- ~CheckedInputStream
  - decaf::util::zip::CheckedInputStream, 1171
- ~CheckedOutputStream
  - decaf::util::zip::CheckedOutputStream, 1173
- ~Checksum
  - decaf::util::zip::Checksum, 1175
- ~ClassCastException
  - decaf::lang::exceptions::ClassCastException, 1179
- ~CloseTransportsTask
  - activemq::transport::failover::CloseTransportsTask, 1182
- ~Closeable
  - cms::Closeable, 1180
  - decaf::io::Closeable, 1181
- ~CmsAccessor
  - activemq::cmsutil::CmsAccessor, 1185
- ~CmsDestinationAccessor
  - activemq::cmsutil::CmsDestinationAccessor, 1189
- ~CmsTemplate
  - activemq::cmsutil::CmsTemplate, 1205
- ~Collection
  - decaf::util::Collection, 1218
- ~Command
  - activemq::commands::Command, 1228
- ~CommandVisitor
  - activemq::state::CommandVisitor, 1235
- ~CommandVisitorAdapter
  - activemq::state::CommandVisitorAdapter, 1245
- ~Comparable
  - decaf::lang::Comparable, 1249
- ~Comparator
  - decaf::util::Comparator, 1252
- ~CompositeData
  - decaf::util::CompositeData, 1254
- ~CompositeTask
  - activemq::threads::CompositeTask, 1255

- ~CompositeTaskRunner
  - activemq::threads::CompositeTaskRunner, 1257
- ~CompositeTransport
  - activemq::transport::CompositeTransport, 1259
- ~ConcurrentMap
  - decaf::util::concurrent::ConcurrentMap, 1261
- ~ConcurrentStlMap
  - decaf::util::concurrent::ConcurrentStlMap, 1270
- ~Condition
  - decaf::util::concurrent::locks::Condition, 1285
- ~ConditionHandle
  - decaf::util::concurrent::ConditionHandle, 1291
- ~ConnectException
  - decaf::net::ConnectException, 1295
- ~Connection
  - cms::Connection, 1298
- ~ConnectionControl
  - activemq::commands::ConnectionControl, 1303
- ~ConnectionControlMarshaller
  - activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller, 1316
  - activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller, 1329
  - activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller, 1308
  - activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller, 1312
  - activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, 1321
  - activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller, 1325
- ~ConnectionError
  - activemq::commands::ConnectionError, 1333
- ~ConnectionErrorMarshaller
  - activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1350
  - activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller, 1337
  - activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller, 1341
  - activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller, 1346
- activemq::wireformat::openwire::marshal::v5::ConnectionError, 1354
- activemq::wireformat::openwire::marshal::v6::ConnectionError, 1358
- ConnectionFactory
  - cms::ConnectionFactory, 1362
- ~ConnectionId
  - activemq::commands::ConnectionId, 1366
- ~ConnectionIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller, 1382
  - activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller, 1370
  - activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller, 1374
  - activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller, 1378
  - activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller, 1386
  - activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller, 1390
- ~ConnectionInfo
  - activemq::commands::ConnectionInfo, 1395
- ~ConnectionInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller, 1414
  - activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller, 1401
  - activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller, 1405
  - activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller, 1409
  - activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller, 1418
  - activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller, 1422
- ~ConnectionMetaData
  - cms::ConnectionMetaData, 1426
- ~ConnectionState
  - activemq::state::ConnectionState, 1431
- ~ConnectionStateMarshaller
  - activemq::wireformat::openwire::marshal::v1::ConnectionStateMarshaller, 1435
- ~ConnectionStateTracker
  - activemq::state::ConnectionStateTracker, 1435
- ~ConsoleHandler
  - decaf::util::logging::ConsoleHandler, 1440
- ~ConsumerControl

activemq::commands::ConsumerControl,	activemq::commands::ControlCommand,
1443	1537
~ConsumerControlMarshaller	~ControlCommandMarshaller
activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller,	activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller,
1461	1553
activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller,	activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller,
1448	1541
activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller,	activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller,
1452	1545
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller,	activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller,
1456	1549
activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller,	activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller,
1465	1558
activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller,	activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller,
1469	1562
~ConsumerId	~CountDownLatch
activemq::commands::ConsumerId,	decaf::util::concurrent::CountDownLatch,
1474	1566
~ConsumerIdMarshaller	
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller,	DataArrayResponse,
1490	activemq::commands::DataArrayResponse,
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller,	1573
1478	~DataArrayResponseMarshaller
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller,	activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller,
1482	1589
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller,	activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller,
1486	1576
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller,	activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller,
1494	1580
activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller,	activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller,
1498	1584
~ConsumerInfo	activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller,
activemq::commands::ConsumerInfo,	1593
1504	activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller,
~ConsumerInfoMarshaller	1597
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller,	DataFormatException,
1524	decaf::util::zip::DataFormatException,
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller,	1602
1511	~DataInput
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller,	decaf::util::zip::DataInput,
1515	1605
~DataInputStream	~DataInputStream
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller,	1615
1519	~DataOutput
activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller,	1624
1528	~DataOutputStream
activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller,	decaf::util::zip::DataOutputStream,
1532	1630
~ConsumerState	~DataResponse
activemq::state::ConsumerState,	activemq::commands::DataResponse,
1536	1633
~ControlCommand	~DataResponseMarshaller

- activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1762
- 1657 ~DeflaterOutputStream
- activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller, 1772
- 1644
- activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller, 1775
- 1649 decaf::util::concurrent::Delayed, 1775
- activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller, 1776
- 1653 cms::DeliveryMode, 1776
- activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller, 1778
- 1636 cms::Destination, 1778
- activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller, 1781
- 1640 activemq::commands::DestinationInfo, 1781
- ~DataStreamMarshaller
- activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller, 1798
- 1661 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller, 1786
- ~DataStructure
- activemq::commands::DataStructure, 1786
- 1714
- ~Date
- decaf::util::Date, 1720
- ~DecafRuntime
- decaf::internal::DecafRuntime, 1724
- ~DedicatedTaskRunner
- activemq::threads::DedicatedTaskRunner, 1807
- 1725
- activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller, 1803
- ~DefaultPrefetchPolicy
- activemq::core::policies::DefaultPrefetchPolicy, 1811
- 1727
- activemq::cmsutil::DestinationResolver, 1811
- ~DefaultRedeliveryPolicy
- activemq::core::policies::DefaultRedeliveryPolicy, 1813
- 1731
- activemq::commands::DiscoveryEvent, 1813
- ~DefaultSSLContext
- decaf::internal::net::ssl::DefaultSSLContext, 1832
- 1744
- activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1820
- ~DefaultSSLServerSocketFactory
- decaf::internal::net::ssl::DefaultSSLServerSocketFactory, 1824
- 1747
- activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, 1824
- ~DefaultSSLSocketFactory
- decaf::internal::net::ssl::DefaultSSLSocketFactory, 1828
- 1753
- activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller, 1836
- ~DefaultServerSocketFactory
- decaf::internal::net::DefaultServerSocketFactory, 1816
- 1736
- activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller, 1816
- ~DefaultSocketFactory
- decaf::internal::net::DefaultSocketFactory, 1841
- 1741 ~Dispatcher
- ~DefaultTransportListener
- activemq::core::Dispatcher, 1841
- activemq::transport::DefaultTransportListener, 1844
- 1758 decaf::lang::Double, 1844
- ~Deflater
- ~DoubleArrayBuffer



- decaf::internal::nio::DoubleArrayBuffer, FileDescriptor, 1859
- decaf::io::FileDescriptor, 1948
- ~DoubleBuffer
  - decaf::nio::DoubleBuffer, 1868
- ~DynamicDestinationResolver
  - activemq::cmsutil::DynamicDestinationResolver, 1879
- ~EOFException
  - decaf::io::EOFException, 1883
- ~Entry
  - decaf::util::Map::Entry, 1881
- ~ErrorManager
  - decaf::util::logging::ErrorManager, 1885
- ~Exception
  - decaf::lang::Exception, 1889
- ~ExceptionListener
  - cms::ExceptionListener, 1894
- ~ExceptionResponse
  - activemq::commands::ExceptionResponse, 1896
- ~ExceptionResponseMarshaller
  - activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller, 1920
  - activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller, 1903
  - activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller, 1907
  - activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller, 1916
  - activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller, 1912
  - activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller, 1899
- ~ExecutionException
  - decaf::util::concurrent::ExecutionException, 1925
- ~Executor
  - decaf::util::concurrent::Executor, 1928
- ~ExecutorService
  - decaf::util::concurrent::ExecutorService, 1929
- ~FailoverTransport
  - activemq::transport::failover::FailoverTransport, 1933
- ~FailoverTransportFactory
  - activemq::transport::failover::FailoverTransportFactory, 1944
- ~FailoverTransportListener
  - activemq::transport::failover::FailoverTransportListener, 1946
- ~Filter
  - decaf::util::logging::Filter, 1950
- ~FilterInputStream
  - decaf::io::FilterInputStream, 1953
- ~FilterOutputStream
  - decaf::io::FilterOutputStream, 1959
- ~Float
  - decaf::lang::Float, 1964
- ~FloatArrayBuffer
  - decaf::internal::nio::FloatArrayBuffer, 1979
- ~FloatBuffer
  - decaf::nio::FloatBuffer, 1988
- ~FlushCommand
  - activemq::commands::FlushCommand, 2000
- ~FlushCommandMarshaller
  - activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller, 2020
  - activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller, 2007
  - activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller, 2011
  - activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller, 2016
  - activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller, 2024
  - activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller, 2003
- ~Flushable
  - decaf::io::Flushable, 1998
- ~Formatter
  - decaf::util::logging::Formatter, 2028
- ~Future
  - decaf::util::concurrent::Future, 2030
- ~FutureResponse
  - activemq::transport::correlator::FutureResponse, 2033
- ~GeneralSecurityException
  - decaf::security::GeneralSecurityException, 2036
- ~GenericResource
  - decaf::internal::util::GenericResource, 2038
- ~Handler
  - decaf::util::logging::Handler, 2043
- ~HexStringParser

- decaf::internal::util::HexStringParser, 2047
- ~HexTable
  - activemq::wireformat::openwire::utils::HexTable, 2048
- ~HttpRetryException
  - decaf::net::HttpRetryException, 2051
- ~IOException
  - decaf::io::IOException, 2212
- ~IOTransport
  - activemq::transport::IOTransport, 2215
- ~IdGenerator
  - activemq::util::IdGenerator, 2053
- ~IllegalArgumentException
  - decaf::lang::exceptions::IllegalArgumentException, 2056
- ~IllegalMonitorStateException
  - decaf::lang::exceptions::IllegalMonitorStateException, 2059
- ~IllegalStateException
  - cms::IllegalStateException, 2060
  - decaf::lang::exceptions::IllegalStateException, 2062
- ~IllegalThreadStateException
  - decaf::lang::exceptions::IllegalThreadStateException, 2065
- ~InactivityMonitor
  - activemq::transport::inactivity::InactivityMonitor, 2067
- ~IndexOutOfBoundsException
  - decaf::lang::exceptions::IndexOutOfBoundsException, 2071
- ~Inet4Address
  - decaf::net::Inet4Address, 2073
- ~Inet6Address
  - decaf::net::Inet6Address, 2077
- ~InetAddress
  - decaf::net::InetAddress, 2080
- ~InetSocketAddress
  - decaf::net::InetSocketAddress, 2085
- ~Inflater
  - decaf::util::zip::Inflater, 2090
- ~InflaterInputStream
  - decaf::util::zip::InflaterInputStream, 2101
- ~InputStream
  - decaf::io::InputStream, 2107
- ~InputStreamReader
  - decaf::io::InputStreamReader, 2118
- ~IntArrayBuffer
  - decaf::internal::nio::IntArrayBuffer, 2125
- ~IntBuffer
  - decaf::nio::IntBuffer, 2133
- ~Integer
  - decaf::lang::Integer, 2146
- ~IntegerResponse
  - activemq::commands::IntegerResponse, 2161
- ~IntegerResponseMarshaller
  - activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller, 2181
  - activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller, 2168
  - activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, 2172
  - activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller, 2177
  - activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller, 2185
  - activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller, 2164
- ~InternalCommandListener
  - activemq::transport::mock::InternalCommandListener, 2193
- ~InterruptedException
  - decaf::lang::exceptions::InterruptedException, 2196
- ~InterruptedIOException
  - decaf::io::InterruptedIOException, 2198
- ~InvalidClientIdException
  - cms::InvalidClientIdException, 2200
- ~InvalidDestinationException
  - cms::InvalidDestinationException, 2201
- ~InvalidKeyException
  - decaf::security::InvalidKeyException, 2203
- ~InvalidMarkException
  - decaf::nio::InvalidMarkException, 2206
- ~InvalidSelectorException
  - cms::InvalidSelectorException, 2207
- ~InvalidStateException
  - decaf::lang::exceptions::InvalidStateException, 2210
- ~Iterable
  - decaf::lang::Iterable, 2221
- ~Iterator
  - decaf::util::Iterator, 2223
- ~JournalQueueAck
  - activemq::commands::JournalQueueAck, 2225
- ~JournalQueueAckMarshaller

- activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller, 2316
- 2249 activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller
- activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller, 2320
- 2233 activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller
- activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller, 2328
- 2241 activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller
- activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller, 2334
- 2245 activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller
- activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller, 2342
- 2237 ~KeepAliveInfo
- activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller, 2348
- 2229 ~KeepAliveInfoMarshaller
- 2336
- ~JournalTopicAck ~KeepAliveInfoMarshaller
- activemq::commands::JournalTopicAck, activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller, 2253 2361
- ~JournalTopicAckMarshaller activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller, 2344
- activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller, 2278
- activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller, 2348
- activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller, 2262
- activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller, 2352
- activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller, 2266
- activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller, 2356
- activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller, 2274
- activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller, 2360
- activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller, 2258
- ~Key
- activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller, 2270
- ~KeyException
- decaf::security::KeyException, 2368
- ~JournalTrace 2282 KeyManagementException
- activemq::commands::JournalTrace, 2282
- ~JournalTraceMarshaller decaf::security::KeyManagementException, 2371
- activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller, 2301
- ~LastPartialCommand
- activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller, PartialCommand, 2285
- 2372
- activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller, LastPartialCommandMarshaller, 2289
- activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller, 2395
- activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller, 2297
- activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller, 2383
- activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller, 2305
- activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller, 2379
- activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller, 2293
- activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller, 2392
- ~JournalTransaction 2392
- activemq::commands::JournalTransaction, activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller, 2309 2388
- ~JournalTransactionMarshaller activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller, 2375
- 2332 ~Less
- activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller, 2400

- ~Level
  - decaf::util::logging::Level, 2405
- ~List
  - decaf::util::List, 2410
- ~ListIterator
  - decaf::util::ListIterator, 2418
- ~LocalTransactionId
  - activemq::commands::LocalTransactionId, 2422
- ~LocalTransactionIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller, 2447
  - activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller, 2430
  - activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller, 2434
  - activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller, 2443
  - activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller, 2439
  - activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller, 2426
- ~Lock
  - decaf::util::concurrent::Lock, 2451
  - decaf::util::concurrent::locks::Lock, 2454
- ~LockSupport
  - decaf::util::concurrent::locks::LockSupport, 2459
- ~LogManager
  - decaf::util::logging::LogManager, 2483
- ~LogRecord
  - decaf::util::logging::LogRecord, 2489
- ~LogWriter
  - decaf::util::logging::LogWriter, 2494
- ~Logger
  - decaf::util::logging::Logger, 2465
- ~LoggerHierarchy
  - decaf::util::logging::LoggerHierarchy, 2474
- ~LoggingInputStream
  - activemq::io::LoggingInputStream, 2475
- ~LoggingOutputStream
  - activemq::io::LoggingOutputStream, 2476
- ~LoggingTransport
  - activemq::transport::logging::LoggingTransport, 2478
- ~Long
  - decaf::lang::Long, 2498
- ~LongArrayBuffer
  - decaf::internal::nio::LongArrayBuffer, 2516
- ~LongBuffer
  - decaf::nio::LongBuffer, 2525
- ~LongSequenceGenerator
  - activemq::util::LongSequenceGenerator, 2535
- ~MalformedURLException
  - decaf::net::MalformedURLException, 2538
- ~Map
  - decaf::util::Map, 2540
- ~MapMessage
  - cms::MapMessage, 2554
- ~MarkBlockLogger
  - decaf::util::logging::MarkBlockLogger, 2564
- ~MarshalAware
  - activemq::wireformat::MarshalAware, 2565
- ~MarshallerFactory
  - activemq::wireformat::openwire::marshal::v1::MarshallerFactory, 2570
  - activemq::wireformat::openwire::marshal::v2::MarshallerFactory, 2571
  - activemq::wireformat::openwire::marshal::v3::MarshallerFactory, 2568
  - activemq::wireformat::openwire::marshal::v4::MarshallerFactory, 2569
  - activemq::wireformat::openwire::marshal::v5::MarshallerFactory, 2569
  - activemq::wireformat::openwire::marshal::v6::MarshallerFactory, 2567
- ~MarshallingSupport
  - activemq::util::MarshallingSupport, 2572
- ~Math
  - decaf::lang::Math, 2578
- ~MemoryUsage
  - activemq::util::MemoryUsage, 2594
- ~Message
  - activemq::commands::Message, 2601
  - cms::Message, 2619
- ~MessageAck
  - activemq::commands::MessageAck, 2644
- ~MessageAckMarshaller
  - activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller, 2667
  - activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller, 2654

- activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller, 2747
- 2658 ~MessageFormatException
- activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller, 2750
- 2662 ~MessageId
- activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller, 2752
- 2671 ~MessageIdMarshaller
- activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller, 2776
- 2650 ~MessageConsumer
- activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 2756
- cms::MessageConsumer, 2675
- ~MessageCreator
- activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller, 2768
- activemq::cmsutil::MessageCreator, 2678
- ~MessageDispatch
- activemq::commands::MessageDispatch, 2760
- 2680
- ~MessageDispatchChannel
- activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller, 2764
- activemq::core::MessageDispatchChannel, 2772
- 2685
- ~MessageDispatchMarshaller
- activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2709
- ~MessageListener
- cms::MessageListener, 2780
- activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller, 2799
- ~MessageMarshaller
- activemq::wireformat::openwire::marshal::v1::MessageMarshaller, 2692
- 2799
- activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller, 2696
- activemq::wireformat::openwire::marshal::v2::MessageMarshaller, 2790
- activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller, 2704
- activemq::wireformat::openwire::marshal::v3::MessageMarshaller, 2786
- activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller, 2700
- activemq::wireformat::openwire::marshal::v4::MessageMarshaller, 2795
- activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller, 2713
- activemq::wireformat::openwire::marshal::v5::MessageMarshaller, 2781
- ~MessageDispatchNotification
- activemq::wireformat::openwire::marshal::v6::MessageMarshaller, 2804
- activemq::commands::MessageDispatchNotification, 2717
- ~MessageDispatchNotificationMarshaller
- ~MessageNotReadableException
- cms::MessageNotReadableException, 2808
- activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller, 2739
- ~MessageNotWriteableException
- cms::MessageNotWriteableException, 2809
- activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller, 2726
- ~MessageProducer
- activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller, 2730
- cms::MessageProducer, 2811
- activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller, 2734
- ~MessagePropertyInterceptor
- activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller, 2743
- activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller, 2819
- activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller, 2722
- activemq::commands::MessagePull, 2825
- ~MessageEOFException
- ~MessagePullMarshaller
- cms::MessageEOFException, 2748
- activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller, 2847
- ~MessageEnumeration

- activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller, 2830
- activemq::wireformat::openwire::marshal::v2::MessagePullMarshallerProviderException, 2915
- activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller, 2838
- decaf::lang::exceptions::NullPointerException, 2917
- activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller, 2842
- ~Number, 2919
- activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller, 2834
- ~NumberFormatException, 2920
- activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller, 2851
- ~NumberFormatException, 2923
- ~MockTransport, 2857
- ~ObjectMessage, 2924
- activemq::transport::mock::MockTransport, cms::ObjectMessage, 2924
- ~MockTransportFactory, 2865
- ~OpenSSLContextSpi, 2926
- decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2865
- ~Mutex, 2868
- decaf::util::concurrent::Mutex, 2868
- ~MutexHandle, 2872
- decaf::util::concurrent::MutexHandle, 2872
- ~Network, 2876
- decaf::internal::net::Network, 2876
- ~NetworkBridgeFilter, 2938
- activemq::commands::NetworkBridgeFilter, 2878
- ~OpenSSLSocket, 2946
- decaf::internal::net::ssl::openssl::OpenSSLSocket, 2946
- ~NetworkBridgeFilterMarshaller, 2902
- activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller, 2902
- decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2958
- activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller, 2882
- ~OpenSSLSocketFactory, 2962
- activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller, 2894
- decaf::internal::net::ssl::openssl::OpenSSLSocketFactory, 2962
- activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller, 2898
- decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2967
- activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller, 2890
- ~OpenSSLSocketOutputStream, 2970
- activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller, 2886
- decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream, 2970
- ~NoRouteToHostException, 2907
- decaf::net::NoRouteToHostException, 2907
- ~OpenWireFormat, 2974
- activemq::wireformat::openwire::OpenWireFormat, 2974
- ~NoSuchAlgorithmException, 2909
- decaf::security::NoSuchAlgorithmException, 2909
- ~OpenWireFormatFactory, 2985
- activemq::wireformat::openwire::OpenWireFormatFactory, 2985
- ~NoSuchElementException, 2912
- decaf::lang::exceptions::NoSuchElementException, 2912
- ~OpenWireFormatNegotiator, 2987
- activemq::wireformat::openwire::OpenWireFormatNegotiator, 2987
- ~NoSuchProviderException, 2987
- ~OpenWireResponseBuilder, 2987

- activemq::wireformat::openwire::OpenWireResponseBuilder, 3115
- 2990
- ~OutputStream
  - decaf::io::OutputStream, 2993
- ~OutputStreamWriter
  - decaf::io::OutputStreamWriter, 3001
- ~PartialCommand
  - activemq::commands::PartialCommand, 3003
- ~PartialCommandMarshaller
  - activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller, 3029
  - activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller, 3011
  - activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller, 3020
  - activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller, 3025
  - activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, 3016
  - activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, 3007
- ~Pointer
  - decaf::lang::Pointer, 3036
- ~PooledSession
  - activemq::cmsutil::PooledSession, 3045
- ~PooledThread
  - decaf::util::concurrent::PooledThread, 3057
- ~PooledThreadListener
  - decaf::util::concurrent::PooledThreadListener, 3059
- ~PortUnreachableException
  - decaf::net::PortUnreachableException, 3062
- ~PrefetchPolicy
  - activemq::core::PrefetchPolicy, 3064
- ~PrimitiveList
  - activemq::util::PrimitiveList, 3070
- ~PrimitiveMap
  - activemq::util::PrimitiveMap, 3082
- ~PrimitiveTypesMarshaller
  - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3092
- ~PrimitiveValueConverter
  - activemq::util::PrimitiveValueConverter, 3099
- ~PrimitiveValueNode
  - activemq::util::PrimitiveValueNode, 3107
- ~Principal
  - activemq::wireformat::openwire::marshal::v1::Principal, 3115
  - ~PriorityQueue
    - decaf::util::PriorityQueue, 3119
  - ~ProducerAck
    - activemq::commands::ProducerAck, 3126
  - ~ProducerAckMarshaller
    - activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller, 3151
    - activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller, 3130
    - activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller, 3138
    - activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller, 3134
    - activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller, 3142
    - activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller, 3147
  - ~ProducerCallback
    - activemq::cmsutil::ProducerCallback, 3156
  - ~ProducerExecutor
    - activemq::cmsutil::CmsTemplate::ProducerExecutor, 3156
  - ~ProducerId
    - activemq::commands::ProducerId, 3158
  - ~ProducerIdMarshaller
    - activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller, 3182
    - activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller, 3162
    - activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller, 3170
    - activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller, 3166
    - activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller, 3174
    - activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, 3178
  - ~ProducerInfo
    - activemq::commands::ProducerInfo, 3187
  - ~ProducerInfoMarshaller
    - activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller, 3200
    - activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller, 3196
    - activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller, 3208
    - activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller, 3191

- activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller, 3285
- 3204 ~RemoveInfoMarshaller
- activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller, 3302
- 3213 ~RemoveInfoMarshaller
- ~ProducerState
- activemq::state::ProducerState, 3216
- ~Properties
- decaf::util::Properties, 3219
- ~PropertiesChangeListener
- decaf::util::logging::PropertiesChangeListener, 3310
- 3227
- ~ProtocolException
- decaf::net::ProtocolException, 3230
- ~PublicKey
- decaf::security::PublicKey, 3231
- ~PushbackInputStream
- decaf::io::PushbackInputStream, 3234
- ~Queue
- cms::Queue, 3239
- decaf::util::Queue, 3241
- ~QueueBrowser
- cms::QueueBrowser, 3244
- ~ReadChecker
- activemq::transport::inactivity::ReadChecker, 3253
- ~ReadOnlyBufferException
- decaf::nio::ReadOnlyBufferException, 3262
- ~ReadWriteLock
- decaf::util::concurrent::locks::ReadWriteLock, 3265
- ~Readable
- decaf::lang::Readable, 3252
- ~Reader
- decaf::io::Reader, 3255
- ~ReceiveExecutor
- activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3266
- ~RedeliveryPolicy
- activemq::core::RedeliveryPolicy, 3269
- ~ReentrantLock
- decaf::util::concurrent::locks::ReentrantLock, 3275
- ~RejectedExecutionException
- decaf::util::concurrent::RejectedExecutionException, 3282
- ~RejectedExecutionHandler
- decaf::util::concurrent::RejectedExecutionHandler, 3283
- ~RemoveInfo
- ~RemoveSubscriptionInfo
- activemq::commands::RemoveSubscriptionInfo, 3315
- ~RemoveSubscriptionInfoMarshaller
- activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller, 3319
- activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller, 3328
- activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller, 3323
- activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller, 3340
- activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller, 3336
- activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller, 3332
- ~ReplayCommand
- activemq::commands::ReplayCommand, 3345
- ~ReplayCommandMarshaller
- activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller, 3352
- activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller, 3357
- activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller, 3361
- activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller, 3348
- activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller, 3369
- activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller, 3365
- ~ResolveProducerExecutor
- activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 3373
- ~ResolveReceiveExecutor



- activemq::cmsutil::CmsTemplate::ResolveSSLContextFactory, 3374
- decaf::net::ssl::SSLContextFactory, 3680
- ~Resource
  - decaf::internal::util::Resource, 3375
- ~ResourceLifecycleManager
  - activemq::cmsutil::ResourceLifecycleManager, 3377
  - decaf::internal::util::ResourceLifecycleManager, 3376
- ~Response
  - activemq::commands::Response, 3380
- ~ResponseBuilder
  - activemq::transport::mock::ResponseBuilder, 3383
- ~ResponseCorrelator
  - activemq::transport::correlator::ResponseCorrelator, 3385
- ~ResponseMarshaller
  - activemq::wireformat::openwire::marshal::SessionResponseMarshaller, 3410
  - activemq::wireformat::openwire::marshal::SessionResponseMarshaller, 3395
  - activemq::wireformat::openwire::marshal::SessionResponseMarshaller, 3405
  - activemq::wireformat::openwire::marshal::SessionResponseMarshaller, 3390
  - activemq::wireformat::openwire::marshal::v5::ResponseMarshaller, 3400
  - activemq::wireformat::openwire::marshal::v6::ResponseMarshaller, 3415
- ~Runnable
  - decaf::lang::Runnable, 3419
- ~Runtime
  - decaf::lang::Runtime, 3420
- ~RuntimeException
  - decaf::lang::exceptions::RuntimeException, 3423
- ~SSLContext
  - decaf::net::ssl::SSLContext, 3654
- ~SSLContextSpi
  - decaf::net::ssl::SSLContextSpi, 3656
- ~SSLParameters
  - decaf::net::ssl::SSLParameters, 3660
- ~SSLServerSocket
  - decaf::net::ssl::SSLServerSocket, 3665
- ~SSLServerSocketFactory
  - decaf::net::ssl::SSLServerSocketFactory, 3669
- ~SSLSocket
  - decaf::net::ssl::SSLSocket, 3674
- ~SecureRandom
  - decaf::security::SecureRandom, 3426
- ~SecureRandomImpl
  - decaf::internal::security::SecureRandomImpl, 3430
- ~SecureRandomSpi
  - decaf::security::SecureRandomSpi, 3433
- ~Semaphore
  - decaf::util::concurrent::Semaphore, 3438
- ~SendExecutor
  - activemq::cmsutil::CmsTemplate::SendExecutor, 3446
- ~ServerSocket
  - decaf::net::ServerSocket, 3451
- ~ServerSocketFactory
  - decaf::net::ServerSocketFactory, 3458
- ~Session
  - cms::Session, 3464
- ~SessionCallback
  - activemq::cmsutil::SessionCallback, 3476
- ~SessionId
  - activemq::commands::SessionId, 3478
- ~SessionIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller, 3501
  - activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller, 3488
  - activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller, 3498
  - activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller, 3486
  - activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller, 3494
  - activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller, 3490
- ~SessionInfo
  - activemq::commands::SessionInfo, 3506
- ~SessionInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller, 3518
  - activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller, 3527
  - activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller, 3522
  - activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller, 3531
  - activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller, 3514

- activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 3638
- 3510
- ~SocketImplFactory
- decaf::net::SocketImplFactory, 3645
- ~SessionPool
- activemq::cmsutil::SessionPool, 3535
- ~SocketOptions
- decaf::net::SocketOptions, 3647
- ~SessionState
- activemq::state::SessionState, 3537
- ~SocketTimeoutException
- decaf::net::SocketTimeoutException, 3652
- ~Set
- decaf::util::Set, 3538
- ~Short
- decaf::lang::Short, 3541
- ~ShortArrayBuffer
- decaf::internal::nio::ShortArrayBuffer, 3554
- ~SslTransport
- activemq::transport::tcp::SslTransport, 3683
- ~SslTransportFactory
- activemq::transport::tcp::SslTransportFactory, 3685
- ~ShortBuffer
- decaf::nio::ShortBuffer, 3563
- ~StandardErrorOutputStream
- decaf::internal::io::StandardErrorOutputStream, 3687
- ~ShutdownInfo
- activemq::commands::ShutdownInfo, 3574
- ~StandardInputStream
- decaf::internal::io::StandardInputStream, 3689
- ~ShutdownInfoMarshaller
- activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller, 3586
- ~StandardOutputStream
- decaf::internal::io::StandardOutputStream, 3690
- activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller, 3581
- ~StaticStartable
- activemq::wireformat::openwire::marshal::StaticStartable, 3594
- ~StaticShutdownInfoMarshaller
- activemq::wireformat::openwire::marshal::StaticShutdownInfoMarshaller, 3598
- ~StaticInitializer
- activemq::core::ActiveMQConstants::StaticInitializer, 3693
- activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller, 3590
- ~StlList
- activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller, 3577
- ~StlMap
- decaf::util::StlMap, 3713
- ~SignatureException
- decaf::security::SignatureException, 3605
- ~StlQueue
- decaf::util::StlQueue, 3725
- ~SimpleFormatter
- decaf::util::logging::SimpleFormatter, 3605
- ~StlSet
- decaf::util::StlSet, 3734
- ~SimpleLogger
- decaf::util::logging::SimpleLogger, 3606
- ~StompFrame
- activemq::wireformat::stomp::StompFrame, 3742
- ~Socket
- decaf::net::Socket, 3614
- ~StompHelper
- activemq::wireformat::stomp::StompHelper, 3747
- ~SocketAddress
- decaf::net::SocketAddress, 3626
- ~StompWireFormat
- activemq::wireformat::stomp::StompWireFormat, 3752
- ~SocketException
- decaf::net::SocketException, 3629
- ~StompWireFormatFactory
- activemq::wireformat::stomp::StompWireFormatFactory, 3755
- ~SocketFactory
- decaf::net::SocketFactory, 3631
- ~Stoppable
- cms::Stoppable, 3756
- ~SocketFileDescriptor
- decaf::internal::net::SocketFileDescriptor, 3635
- ~SocketImpl

- ~StreamHandler
  - decaf::util::logging::StreamHandler, 3758
- ~StreamMessage
  - cms::StreamMessage, 3763
- ~String
  - decaf::lang::String, 3777
- ~StringTokenizer
  - decaf::util::StringTokenizer, 3780
- ~SubscriptionInfo
  - activemq::commands::SubscriptionInfo, 3783
- ~SubscriptionInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller, 3792
  - activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller, 3808
  - activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller, 3788
  - activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller, 3800
  - activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller, 3796
  - activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller, 3804
- ~Synchronizable
  - decaf::util::concurrent::Synchronizable, 3812
- ~SynchronizableImpl
  - decaf::internal::util::concurrent::SynchronizableImpl, 3823
- ~Synchronization
  - activemq::core::Synchronization, 3827
- ~SynchronousQueue
  - decaf::util::concurrent::SynchronousQueue, 3830
- ~System
  - decaf::lang::System, 3840
- ~Task
  - activemq::threads::Task, 3847
- ~TaskListener
  - decaf::util::concurrent::TaskListener, 3848
- ~TaskRunner
  - activemq::threads::TaskRunner, 3849
- ~TcpSocket
  - decaf::internal::net::tcp::TcpSocket, 3854
- ~TcpSocketInputStream
  - decaf::internal::net::tcp::TcpSocketInputStream, 3861
- ~TcpSocketOutputStream
  - decaf::internal::net::tcp::TcpSocketOutputStream, 3864
- ~TcpTransport
  - activemq::transport::tcp::TcpTransport, 3866
- ~TcpTransportFactory
  - activemq::transport::tcp::TcpTransportFactory, 3870
- ~TemporaryQueue
  - cms::TemporaryQueue, 3872
- ~TemporaryTopic
  - cms::TemporaryTopic, 3873
- ~TextMessage
  - cms::TextMessage, 3875
- ~Thread
  - decaf::lang::Thread, 3881
- ~ThreadFactory
  - decaf::util::concurrent::ThreadFactory, 3887
- ~ThreadGroup
  - decaf::lang::ThreadGroup, 3888
- ~ThreadPool
  - decaf::util::concurrent::ThreadPool, 3891
- ~Throwable
  - decaf::lang::Throwable, 3896
- ~TimeUnit
  - decaf::util::concurrent::TimeUnit, 3922
- ~TimeoutException
  - decaf::util::concurrent::TimeoutException, 3901
- ~Timer
  - decaf::util::Timer, 3904
- ~TimerTask
  - decaf::util::TimerTask, 3915
- ~TimerTaskHeap
  - decaf::internal::util::TimerTaskHeap, 3918
- ~Topic
  - cms::Topic, 3931
- ~Tracked
  - activemq::state::Tracked, 3932
- ~TransactionId
  - activemq::commands::TransactionId, 3933
- ~TransactionIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller, 3941
  - activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller, 3945

- activemq::wireformat::openwire::marshal::URIHelper, 3949
- activemq::wireformat::openwire::marshal::URIPool, 3953
- activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller, 3937
- activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller, 3957
- ~TransactionInfo, 3961
- activemq::commands::TransactionInfo, ~URL, 3961
- ~TransactionInfoMarshaller, 3969
- activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller, 3969
- activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller, 3986
- activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller, 3974
- activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller, 3982
- activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller, 3965
- activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller, 3978
- ~TransactionState, 3991
- activemq::state::TransactionState, 3991
- ~TransferQueue, 3993
- decaf::internal::util::concurrent::TransferQueue, 3993
- ~TransferStack, 3995
- decaf::internal::util::concurrent::TransferStack, 3995
- ~Transport, 3998
- activemq::transport::Transport, 3998
- ~TransportFactory, 4003
- activemq::transport::TransportFactory, 4003
- ~TransportFilter, 4007
- activemq::transport::TransportFilter, 4007
- ~TransportListener, 4014
- activemq::transport::TransportListener, 4014
- ~TransportRegistry, 4016
- activemq::transport::TransportRegistry, 4016
- ~URI, 4036
- decaf::net::URI, 4036
- ~URIEncoderDecoder, 4045
- decaf::internal::net::URIEncoderDecoder, 4045
- URIHelper, 3949
- decaf::internal::net::URIHelper, 4049
- URIPool, 3953
- activemq::transport::failover::URIPool, 4056
- TransactionIdMarshaller, 3937
- ~URISyntaxException, 4063
- decaf::net::URISyntaxException, 4063
- ~URITYPE, 4066
- decaf::internal::net::URITYPE, 4066
- ~URL, 4073
- decaf::net::URL, 4073
- ~URLDecoder, 4074
- decaf::net::URLDecoder, 4074
- ~URLEncoder, 4075
- decaf::net::URLEncoder, 4075
- ~UTFDataFormatException, 4080
- decaf::io::UTFDataFormatException, 4080
- ~UUID, 4083
- decaf::util::UUID, 4083
- ~UncaughtExceptionHandler, 4019
- decaf::lang::Thread::UncaughtExceptionHandler, 4019
- ~UnknownHostException, 4022
- decaf::net::UnknownHostException, 4022
- ~UnknownServiceException, 4024
- decaf::net::UnknownServiceException, 4024
- ~UnsupportedEncodingException, 4027
- decaf::io::UnsupportedEncodingException, 4027
- ~UnsupportedOperationException, 4031
- cms::UnsupportedOperationException, 4031
- decaf::lang::exceptions::UnsupportedOperationException, 4030
- ~Usage, 4076
- activemq::util::Usage, 4076
- ~WireFormat, 4089
- activemq::wireformat::WireFormat, 4089
- ~WireFormatFactory, 4093
- activemq::wireformat::WireFormatFactory, 4093
- ~WireFormatInfo, 4096
- activemq::commands::WireFormatInfo, 4096
- ~WireFormatInfoMarshaller, 4122
- activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller, 4122

---

activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller, 4114  
     \_array  
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller, 4126  
     1148  
 activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller, 4118  
     decaf::nio::Buffer, 942  
 activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller, 4106  
     deflate.h, 4623  
 activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller, 4110  
     \_length\_code  
 ~WireFormatNegotiator  
     deflate.h, 4623  
 activemq::wireformat::WireFormatNegotiator, 4129  
     \_limit  
 ~WireFormatRegistry  
     decaf::nio::Buffer, 942  
 activemq::wireformat::WireFormatRegistry, 4131  
     \_mark  
 ~WriteChecker  
     decaf::nio::Buffer, 942  
     markSet  
 activemq::transport::inactivity::WriteChecker, 4133  
     decaf::nio::Buffer, 942  
     \_position  
 ~Writer  
     decaf::io::Writer, 4135  
     \_tr\_tally\_dist  
 ~X500Principal  
     decaf::security::auth::x500::X500Principal, 4140  
     \_tr\_tally\_lit  
     deflate.h, 4622  
 ~X509Certificate  
     decaf::security::cert::X509Certificate, 4142  
     ABORT  
 ~XATransactionId  
     activemq::wireformat::stomp::StompCommandConstants, 3740  
     abs  
 ~XATransactionIdMarshaller  
     decaf::lang::Math, 2578, 2579  
     AbstractCollection  
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller, 4161  
     decaf::util::AbstractCollection, 162  
     AbstractQueue  
 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller, 4152  
     decaf::util::AbstractQueue, 177  
     accept  
 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller, 4165  
     decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2932  
 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller, 4157  
     decaf::internal::net::tcp::TcpSocket, 3854  
     decaf::net::ServerSocket, 3451  
 activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller, 4169  
     decaf::net::SocketImpl, 3638  
     hideGet  
 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller, 4148  
     decaf::net::Socket, 3614  
 ~XMLFormatter  
     ACK  
     decaf::util::logging::XMLFormatter, 4173  
     activemq::wireformat::stomp::StompCommandConstants, 3740  
 ~ZipException  
     decaf::util::zip::ZipException, 4177  
     ACK\_AUTO  
 \_FALSE  
     activemq::wireformat::stomp::StompCommandConstants, 3740  
     decaf::lang::Boolean, 861  
 \_TRUE  
     ACK\_CLIENT

---

activemq::wireformat::stomp::StompCommand, 3740  
 ACK\_INDIVIDUAL  
 activemq::wireformat::stomp::StompCommand, 3740  
 ACK\_TYPE\_CONSUMED  
 activemq::core::ActiveMQConstants, 298  
 ACK\_TYPE\_DELIVERED  
 activemq::core::ActiveMQConstants, 298  
 ACK\_TYPE\_INDIVIDUAL  
 activemq::core::ActiveMQConstants, 298  
 ACK\_TYPE\_POISON  
 activemq::core::ActiveMQConstants, 298  
 ACK\_TYPE\_REDELIVERED  
 activemq::core::ActiveMQConstants, 298  
 acknowledge  
 activemq::commands::ActiveMQMessageTemplate, 422  
 activemq::core::ActiveMQConsumer, 303  
 activemq::core::ActiveMQSession, 517  
 cms::Message, 2619  
 acknowledgeMessage  
 activemq::core::ActiveMQAckHandler, 184  
 AcknowledgeMode  
 cms::Session, 3464  
 AckType  
 activemq::core::ActiveMQConstants, 298  
 ackType  
 activemq::commands::MessageAck, 2648  
 acquire  
 decaf::util::concurrent::Semaphore, 3438  
 acquireUninterruptibly  
 decaf::util::concurrent::Semaphore, 3439, 3440  
 action  
 activemq::cmsutil::CmsTemplate::Producer, 3156  
 activemq, 93  
 activemq/exceptions/ExceptionDefines.h  
 AMQ\_CATCH\_EXCEPTION\_CONVERT, 4238  
 AMQ\_CATCH\_NOTHROW, 4239  
 AMQ\_CATCH\_RETHROW, 4239  
 AMQ\_CATCHALL\_NOTHROW, 4239  
 AMQ\_CATCHALL\_THROW, 4240  
 activemq::config.h  
 AMQCPP\_API, 4275  
 HAVE\_PTHREAD\_H, 4275  
 HAVE\_UUID\_T, 4275  
 HAVE\_UUID\_UUID\_H, 4275  
 activemq::cmsutil, 94  
 activemq::cmsutil::CachedConsumer, 1097  
 ~CachedConsumer, 1099  
 CachedConsumer, 1099  
 close, 1099  
 getMessageListener, 1099  
 getMessageSelector, 1099  
 operator=, 1100  
 receive, 1100  
 receiveNoWait, 1100  
 setMessageListener, 1101  
 activemq::cmsutil::CachedProducer, 1101  
 ~CachedProducer, 1103  
 CachedProducer, 1103  
 close, 1103  
 getDeliveryMode, 1103  
 getDisableMessageID, 1103  
 getDisableMessageTimeStamp, 1104  
 getPriority, 1104  
 getTimeToLive, 1104  
 operator=, 1105  
 send, 1105, 1106  
 setDeliveryMode, 1107  
 setDisableMessageID, 1107  
 setDisableMessageTimeStamp, 1107  
 setPriority, 1108  
 setTimeToLive, 1108  
 activemq::cmsutil::CmsAccessor, 1183  
 ~CmsAccessor, 1185  
 checkConnectionFactory, 1185  
 CmsAccessor, 1185  
 createConnection, 1185  
 createSession, 1185  
 destroy, 1186  
 getConnectionFactory, 1186  
 getResourceLifecycleManager, 1186  
 getSessionAcknowledgeMode, 1186  
 init, 1186  
 operator=, 1187  
 setConnectionFactory, 1187  
 setSessionAcknowledgeMode, 1187

- activemq::cmsutil::CmsDestinationAccessor, 1187
  - ~CmsDestinationAccessor, 1189
  - checkDestinationResolver, 1189
  - CmsDestinationAccessor, 1189
  - destroy, 1189
  - getDestinationResolver, 1189
  - init, 1189
  - isPubSubDomain, 1189
  - operator=, 1190
  - resolveDestinationName, 1190
  - setDestinationResolver, 1190
  - setPubSubDomain, 1190
- activemq::cmsutil::CmsTemplate, 1201
  - ~CmsTemplate, 1205
  - CmsTemplate, 1205
  - DEFAULT\_PRIORITY, 1215
  - DEFAULT\_TIME\_TO\_LIVE, 1215
  - destroy, 1205
  - execute, 1205, 1206
  - getDefaultDestination, 1206, 1207
  - getDefaultDestinationName, 1207
  - getDeliveryMode, 1207
  - getPriority, 1207
  - getReceiveTimeout, 1207
  - getTimeToLive, 1207
  - init, 1207
  - isExplicitQosEnabled, 1207
  - isMessageIdEnabled, 1208
  - isMessageTimestampEnabled, 1208
  - isNoLocal, 1208
  - operator=, 1208
  - ProducerExecutor, 1215
  - receive, 1208, 1209
  - RECEIVE\_TIMEOUT\_INDEFINITE\_WAIT, 1215
  - RECEIVE\_TIMEOUT\_NO\_WAIT, 1215
  - ReceiveExecutor, 1215
  - receiveSelected, 1209, 1210
  - ResolveProducerExecutor, 1215
  - ResolveReceiveExecutor, 1215
  - send, 1210, 1211
  - SendExecutor, 1215
  - setDefaultDestination, 1212
  - setDefaultDestinationName, 1212
  - setDeliveryMode, 1212
  - setDeliveryPersistent, 1212
  - setExplicitQosEnabled, 1213
  - setMessageIdEnabled, 1213
  - setMessageTimestampEnabled, 1213
  - setNoLocal, 1213
  - setPriority, 1213
  - setPubSubDomain, 1213
  - setReceiveTimeout, 1214
  - setTimeToLive, 1214
- activemq::cmsutil::CmsTemplate::ProducerExecutor, 3155
  - ~ProducerExecutor, 3156
  - action, 3156
  - destination, 3156
  - doInCms, 3156
  - getDestination, 3156
  - operator=, 3156
  - parent, 3156
  - ProducerExecutor, 3156
- activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3265
  - ~ReceiveExecutor, 3266
  - destination, 3267
  - doInCms, 3266
  - getDestination, 3266
  - getMessage, 3267
  - message, 3267
  - noLocal, 3267
  - operator=, 3267
  - parent, 3267
  - ReceiveExecutor, 3266
  - selector, 3267
- activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 3372
  - ~ResolveProducerExecutor, 3373
  - getDestination, 3373
  - operator=, 3373
  - ResolveProducerExecutor, 3373
- activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 3373
  - ~ResolveReceiveExecutor, 3374
  - getDestination, 3374
  - operator=, 3374
  - ResolveReceiveExecutor, 3374
- activemq::cmsutil::CmsTemplate::SendExecutor, 3445
  - ~SendExecutor, 3446
  - doInCms, 3446
  - operator=, 3446
  - SendExecutor, 3446
- activemq::cmsutil::DestinationResolver, 1810
  - ~DestinationResolver, 1811
  - destroy, 1811
  - init, 1811

- resolveDestinationName, 1811
- activemq::cmsutil::DynamicDestinationResolver, operator=, 3379
  - 1878
  - ~DynamicDestinationResolver, 1879
  - destroy, 1879
  - DynamicDestinationResolver, 1879
  - init, 1879
  - operator=, 1879
  - resolveDestinationName, 1880
- activemq::cmsutil::MessageCreator, 2677
  - ~MessageCreator, 2678
  - createMessage, 2678
- activemq::cmsutil::PooledSession, 3041
  - ~PooledSession, 3045
  - close, 3045
  - commit, 3045
  - createBrowser, 3045, 3046
  - createBytesMessage, 3046
  - createCachedConsumer, 3047
  - createCachedProducer, 3047
  - createConsumer, 3048, 3049
  - createDurableConsumer, 3049
  - createMapMessage, 3050
  - createMessage, 3050
  - createProducer, 3051
  - createQueue, 3051
  - createStreamMessage, 3051
  - createTemporaryQueue, 3052
  - createTemporaryTopic, 3052
  - createTextMessage, 3052, 3053
  - createTopic, 3053
  - getAcknowledgeMode, 3053
  - getSession, 3054
  - isTransacted, 3054
  - operator=, 3054
  - PooledSession, 3045
  - recover, 3054
  - rollback, 3055
  - unsubscribe, 3055
- activemq::cmsutil::ProducerCallback, 3154
  - ~ProducerCallback, 3154
  - doInCms, 3154
- activemq::cmsutil::ResourceLifecycleManager, 3376
  - ~ResourceLifecycleManager, 3377
  - addConnection, 3377
  - addDestination, 3377
  - addMessageConsumer, 3378
  - addMessageProducer, 3378
  - addSession, 3378
  - destroy, 3378
  - releaseAll, 3379
  - ResourceLifecycleManager, 3377
- activemq::cmsutil::SessionCallback, 3475
  - ~SessionCallback, 3476
  - doInCms, 3476
- activemq::cmsutil::SessionPool, 3534
  - ~SessionPool, 3535
  - getResourceLifecycleManager, 3535
  - operator=, 3535
  - returnSession, 3535
  - SessionPool, 3535
  - takeSession, 3535
- activemq::commands, 95
- activemq::commands::ActiveMQBlobMessage, 184
  - ~ActiveMQBlobMessage, 186
  - ActiveMQBlobMessage, 186
  - BINARY\_MIME\_TYPE, 189
  - clone, 186
  - cloneDataStructure, 186
  - copyDataStructure, 186
  - equals, 187
  - getDataStructureType, 187
  - getMimeType, 187
  - getName, 187
  - getRemoteBlobUrl, 187
  - ID\_ACTIVEMQBLOBMESSAGE, 189
  - isDeletedByBroker, 188
  - setDeletedByBroker, 188
  - setMimeType, 188
  - setName, 188
  - setRemoteBlobUrl, 188
  - toString, 189
- activemq::commands::ActiveMQBytesMessage, 215
  - ~ActiveMQBytesMessage, 219
  - ActiveMQBytesMessage, 219
  - clearBody, 219
  - clone, 219
  - cloneDataStructure, 219
  - copyDataStructure, 219
  - equals, 220
  - getBodyBytes, 220
  - getBodyLength, 220
  - getDataStructureType, 221
  - ID\_ACTIVEMQBYTESMESSAGE, 234
  - onSend, 221



- readBoolean, 221
- readByte, 222
- readBytes, 222, 223
- readChar, 224
- readDouble, 224
- readFloat, 225
- readInt, 225
- readLong, 225
- readShort, 226
- readString, 226
- readUnsignedShort, 227
- readUTF, 227
- reset, 228
- setBodyBytes, 228
- toString, 229
- writeBoolean, 229
- writeByte, 229
- writeBytes, 230
- writeChar, 231
- writeDouble, 231
- writeFloat, 231
- writeInt, 232
- writeLong, 232
- writeShort, 233
- writeString, 233
- writeUnsignedShort, 233
- writeUTF, 234
- activemq::commands::ActiveMQDestination, 312
  - ~ActiveMQDestination, 315
  - ActiveMQDestination, 315
  - advisory, 322
  - ADVISORY\_PREFIX, 322
  - cloneDataStructure, 315
  - COMPOSITE\_SEPARATOR, 322
  - CONNECTION\_ADVISORY\_PREFIX, 322
  - CONSUMER\_ADVISORY\_PREFIX, 322
  - copyDataStructure, 315
  - createDestination, 316
  - createTemporaryName, 316
  - DEFAULT\_ORDERED\_TARGET, 322
  - equals, 316
  - exclusive, 322
  - getClientId, 317
  - getCMSDestination, 317
  - getDataStructureType, 317
  - getDestinationType, 318
  - getOptions, 318
  - getOrderedTarget, 318
  - getPhysicalName, 318
  - ID\_ACTIVEMQDESTINATION, 323
  - isAdvisory, 319
  - isComposite, 319
  - isConnectionAdvisory, 319
  - isConsumerAdvisory, 319
  - isExclusive, 319
  - isOrdered, 319
  - isProducerAdvisory, 319
  - isQueue, 320
  - isTemporary, 320
  - isTopic, 320
  - isWildcard, 320
  - options, 323
  - ordered, 323
  - orderedTarget, 323
  - physicalName, 323
  - PRODUCER\_ADVISORY\_PREFIX, 323
  - QUEUE\_QUALIFIED\_PREFIX, 323
  - setAdvisory, 320
  - setExclusive, 321
  - setOrdered, 321
  - setOrderedTarget, 321
  - setPhysicalName, 321
  - TEMP\_POSTFIX, 323
  - TEMP\_PREFIX, 323
  - TEMP\_QUEUE\_QUALIFIED\_PREFIX, 323
  - TEMP\_TOPIC\_QUALIFIED\_PREFIX, 323
  - TOPIC\_QUALIFIED\_PREFIX, 323
  - toString, 321
- activemq::commands::ActiveMQDestination::DestinationFilter, 1779
  - ANY\_CHILD, 1779
  - ANY\_DESCENDENT, 1779
- activemq::commands::ActiveMQMapMessage, 350
  - ~ActiveMQMapMessage, 353
  - ActiveMQMapMessage, 353
  - beforeMarshal, 353
  - checkMapIsUnmarshalled, 353
  - clearBody, 353
  - clone, 354
  - cloneDataStructure, 354
  - copyDataStructure, 354
  - equals, 354
  - getBoolean, 355

- getByte, 355
- getBytes, 356
- getChar, 356
- getDataStructureType, 356
- getDouble, 356
- getFloat, 357
- getInt, 357
- getLong, 358
- getMap, 358
- getMapNames, 358
- getShort, 359
- getString, 359
- ID\_ACTIVEMQMAPMESSAGE, 364
- isMarshalAware, 359
- itemExists, 360
- setBoolean, 360
- setByte, 360
- setBytes, 361
- setChar, 361
- setDouble, 362
- setFloat, 362
- setInt, 362
- setLong, 363
- setShort, 363
- setString, 363
- toString, 364
- activemq::commands::ActiveMQMessage, 390
  - ~ActiveMQMessage, 391
  - ActiveMQMessage, 391
  - clone, 391
  - cloneDataStructure, 391
  - copyDataStructure, 391
  - equals, 391
  - getDataStructureType, 392
  - ID\_ACTIVEMQMESSAGE, 392
  - toString, 392
- activemq::commands::ActiveMQMessageTemplate, 418
  - ~ActiveMQMessageTemplate, 422
  - acknowledge, 422
  - ActiveMQMessageTemplate, 422
  - clearBody, 422
  - clearProperties, 422
  - equals, 423
  - failIfReadOnlyBody, 423
  - failIfReadOnlyProperties, 423
  - failIfWriteOnlyBody, 423
  - getBooleanProperty, 423
  - getByteProperty, 424
  - getCMSCorrelationID, 424
  - getCMSDeliveryMode, 424
  - getCMSDestination, 425
  - getCMSExpiration, 425
  - getCMSMessageID, 425
  - getCMSPriority, 426
  - getCMSRedelivered, 426
  - getCMSReplyTo, 426
  - getCMSTimestamp, 427
  - getCMSType, 427
  - getDoubleProperty, 427
  - getFloatProperty, 428
  - getIntProperty, 428
  - getLongProperty, 429
  - getPropertyNames, 429
  - getShortProperty, 430
  - getStringProperty, 430
  - onSend, 431
  - propertyExists, 431
  - setBooleanProperty, 431
  - setByteProperty, 432
  - setCMSCorrelationID, 432
  - setCMSDeliveryMode, 432
  - setCMSDestination, 433
  - setCMSExpiration, 433
  - setCMSMessageID, 433
  - setCMSPriority, 433
  - setCMSRedelivered, 434
  - setCMSReplyTo, 434
  - setCMSTimestamp, 434
  - setCMSType, 435
  - setDoubleProperty, 435
  - setFloatProperty, 435
  - setIntProperty, 436
  - setLongProperty, 436
  - setShortProperty, 437
  - setStringProperty, 437
- activemq::commands::ActiveMQObjectMessage, 438
  - ~ActiveMQObjectMessage, 439
  - ActiveMQObjectMessage, 439
  - clone, 439
  - cloneDataStructure, 439
  - copyDataStructure, 439
  - equals, 439
  - getDataStructureType, 440
  - ID\_ACTIVEMQOBJECTMESSAGE, 440
  - toString, 440
- activemq::commands::ActiveMQQueue, 479

- ~ActiveMQQueue, 480
- ActiveMQQueue, 480
- clone, 480
- cloneDataStructure, 481
- copy, 481
- copyDataStructure, 481
- equals, 481
- getCMSDestination, 482
- getCMSProperties, 482
- getDataStructureType, 482
- getDestinationType, 482
- getQueueName, 483
- ID\_ACTIVEMQQUEUE, 483
- toString, 483
- activemq::commands::ActiveMQStreamMessage
  - 535
  - ~ActiveMQStreamMessage, 539
  - ActiveMQStreamMessage, 539
  - clearBody, 539
  - clone, 539
  - cloneDataStructure, 539
  - copyDataStructure, 539
  - equals, 540
  - getDataStructureType, 540
  - ID\_ACTIVEMQSTREAMMESSAGE, 553
  - onSend, 540
  - readBoolean, 540
  - readByte, 541
  - readBytes, 541, 542
  - readChar, 543
  - readDouble, 544
  - readFloat, 544
  - readInt, 544
  - readLong, 545
  - readShort, 545
  - readString, 546
  - readUnsignedShort, 546
  - reset, 547
  - toString, 547
  - writeBoolean, 547
  - writeByte, 548
  - writeBytes, 548, 549
  - writeChar, 549
  - writeDouble, 550
  - writeFloat, 550
  - writeInt, 550
  - writeLong, 551
  - writeShort, 551
  - writeString, 552
  - writeUnsignedShort, 552
- activemq::commands::ActiveMQTempDestination, 578
  - ~ActiveMQTempDestination, 580
  - ActiveMQTempDestination, 580
  - cloneDataStructure, 580
  - close, 580
  - connection, 582
  - copyDataStructure, 580
  - equals, 581
  - getDataStructureType, 581
  - ID\_ACTIVEMQTEMPDESTINATION, 582
  - setConnection, 581
  - toString, 581
- activemq::commands::ActiveMQTempQueue, 606
  - ~ActiveMQTempQueue, 608
  - ActiveMQTempQueue, 608
  - clone, 608
  - cloneDataStructure, 608
  - copy, 608
  - copyDataStructure, 609
  - destroy, 609
  - equals, 609
  - getCMSDestination, 609
  - getCMSProperties, 610
  - getDataStructureType, 610
  - getDestinationType, 610
  - getQueueName, 610
  - ID\_ACTIVEMQTEMPQUEUE, 611
  - toString, 610
- activemq::commands::ActiveMQTempTopic, 636
  - ~ActiveMQTempTopic, 638
  - ActiveMQTempTopic, 638
  - clone, 638
  - cloneDataStructure, 638
  - copy, 638
  - copyDataStructure, 638
  - destroy, 639
  - equals, 639
  - getCMSDestination, 639
  - getCMSProperties, 639
  - getDataStructureType, 640
  - getDestinationType, 640
  - getTopicName, 640
  - ID\_ACTIVEMQTEMPTOPIC, 641
  - toString, 640

- activemq::commands::ActiveMQTextMessage, 666
  - ~ActiveMQTextMessage, 668
  - ActiveMQTextMessage, 668
  - beforeMarshal, 668
  - clearBody, 668
  - clone, 668
  - cloneDataStructure, 669
  - copyDataStructure, 669
  - equals, 669
  - getDataStructureType, 669
  - getSize, 670
  - getText, 670
  - ID\_ACTIVEMQTEXTMESSAGE, 671
  - setText, 670
  - text, 671
  - toString, 671
- activemq::commands::ActiveMQTopic, 697
  - ~ActiveMQTopic, 698
  - ActiveMQTopic, 698
  - clone, 698
  - cloneDataStructure, 698
  - copy, 699
  - copyDataStructure, 699
  - equals, 699
  - getCMSDestination, 699
  - getCMSProperties, 700
  - getDataStructureType, 700
  - getDestinationType, 700
  - getTopicName, 700
  - ID\_ACTIVEMQTOPIC, 701
  - toString, 700
- activemq::commands::BaseCommand, 764
  - ~BaseCommand, 766
  - BaseCommand, 766
  - copyDataStructure, 766
  - equals, 766
  - getCommandId, 767
  - isBrokerInfo, 767
  - isConnectionInfo, 768
  - isConsumerInfo, 768
  - isKeepAliveInfo, 768
  - isMessage, 768
  - isMessageAck, 768
  - isMessageDispatch, 768
  - isMessageDispatchNotification, 768
  - isProducerAck, 769
  - isProducerInfo, 769
  - isRemoveInfo, 769
  - isRemoveSubscriptionInfo, 769
  - isResponse, 769
  - isResponseRequired, 769
  - isShutdownInfo, 769
  - isTransactionInfo, 770
  - isWireFormatInfo, 770
  - setCommandId, 770
  - setResponseRequired, 770
  - toString, 770
- activemq::commands::BaseDataStructure, 837
  - ~BaseDataStructure, 838
  - afterMarshal, 838
  - afterUnmarshal, 838
  - beforeMarshal, 839
  - beforeUnmarshal, 839
  - copyDataStructure, 839
  - equals, 839
  - getMarshaledForm, 840
  - isMarshalAware, 840
  - setMarshaledForm, 840
  - toString, 841
- activemq::commands::BooleanExpression, 861
  - ~BooleanExpression, 862
  - BooleanExpression, 862
  - cloneDataStructure, 862
  - copyDataStructure, 862
  - equals, 862
  - toString, 863
- activemq::commands::BrokerError, 868
  - ~BrokerError, 870
  - BrokerError, 870
  - cloneDataStructure, 870
  - copyDataStructure, 870
  - getCause, 870
  - getDataStructureType, 871
  - getExceptionClass, 871
  - getMessage, 871
  - getStackTraceElements, 871
  - setCause, 871
  - setExceptionClass, 872
  - setMessage, 872
  - setStackTraceElements, 872
  - visit, 872
- activemq::commands::BrokerError::StackTraceElement, 3685
  - ClassName, 3686
  - FileName, 3686
  - LineNumber, 3686
  - MethodName, 3686

- activemq::commands::BrokerId, 874
  - ~BrokerId, 876
  - BrokerId, 876
  - cloneDataStructure, 876
  - COMPARATOR, 876
  - compareTo, 876
  - copyDataStructure, 876
  - equals, 876, 877
  - getDataStructureType, 877
  - getValue, 877
  - ID\_BROKERID, 878
  - operator<, 877
  - operator=, 877
  - operator==, 877
  - setValue, 877
  - toString, 877
  - value, 878
- activemq::commands::BrokerInfo, 902
  - ~BrokerInfo, 904
  - brokerId, 910
  - BrokerInfo, 904
  - brokerName, 910
  - brokerUploadUrl, 910
  - brokerURL, 910
  - cloneDataStructure, 904
  - connectionId, 910
  - copyDataStructure, 904
  - duplexConnection, 910
  - equals, 905
  - faultTolerantConfiguration, 910
  - getBrokerId, 905, 906
  - getBrokerName, 906
  - getBrokerUploadUrl, 906
  - getBrokerURL, 906
  - getConnectionId, 906
  - getDataStructureType, 906
  - getNetworkProperties, 906, 907
  - getPeerBrokerInfos, 907
  - ID\_BROKERINFO, 910
  - isBrokerInfo, 907
  - isDuplexConnection, 907
  - isFaultTolerantConfiguration, 908
  - isMasterBroker, 908
  - isNetworkConnection, 908
  - isSlaveBroker, 908
  - masterBroker, 910
  - networkConnection, 910
  - networkProperties, 910
  - peerBrokerInfos, 910
  - setBrokerId, 908
  - setBrokerName, 908
  - setBrokerUploadUrl, 908
  - setBrokerURL, 908
  - setConnectionId, 908
  - setDuplexConnection, 908
  - setFaultTolerantConfiguration, 908
  - setMasterBroker, 908
  - setNetworkConnection, 908
  - setNetworkProperties, 908
  - setPeerBrokerInfos, 908
  - setSlaveBroker, 908
  - slaveBroker, 910
  - toString, 908
  - visit, 909
- activemq::commands::Command, 1227
  - ~Command, 1228
  - getCommandId, 1228
  - isBrokerInfo, 1228
  - isConnectionInfo, 1228
  - isConsumerInfo, 1229
  - isKeepAliveInfo, 1229
  - isMessage, 1229
  - isMessageAck, 1229
  - isMessageDispatch, 1229
  - isMessageDispatchNotification, 1229
  - isProducerAck, 1229
  - isProducerInfo, 1230
  - isRemoveInfo, 1230
  - isRemoveSubscriptionInfo, 1230
  - isResponse, 1230
  - isResponseRequired, 1230
  - isShutdownInfo, 1230
  - isTransactionInfo, 1230
  - isWireFormatInfo, 1231
  - setCommandId, 1231
  - setResponseRequired, 1231
  - toString, 1231
  - visit, 1232
- activemq::commands::ConnectionControl, 1301
  - ~ConnectionControl, 1303
  - cloneDataStructure, 1303
  - close, 1306
  - connectedBrokers, 1306
  - ConnectionControl, 1303
  - copyDataStructure, 1303
  - equals, 1303
  - exit, 1306
  - faultTolerant, 1306
  - getConnectedBrokers, 1303, 1304

- getDataSetType, 1304
- getReconnectTo, 1304, 1305
- ID\_CONNECTIONCONTROL, 1306
- isClose, 1305
- isExit, 1305
- isFaultTolerant, 1305
- isRebalanceConnection, 1305
- isResume, 1305
- isSuspend, 1305
- rebalanceConnection, 1306
- reconnectTo, 1306
- resume, 1306
- setClose, 1305
- setConnectedBrokers, 1305
- setExit, 1305
- setFaultTolerant, 1305
- setRebalanceConnection, 1305
- setReconnectTo, 1305
- setResume, 1305
- setSuspend, 1305
- suspend, 1306
- toString, 1305
- visit, 1306
- activemq::commands::ConnectionError, 1332
  - ~ConnectionError, 1333
  - cloneDataSet, 1333
  - ConnectionError, 1333
  - connectionId, 1336
  - copyDataSet, 1334
  - equals, 1334
  - exception, 1336
  - getConnectionId, 1334
  - getDataSetType, 1334
  - getException, 1334, 1335
  - ID\_CONNECTIONERROR, 1336
  - setConnectionId, 1335
  - setException, 1335
  - toString, 1335
  - visit, 1335
- activemq::commands::ConnectionId, 1365
  - ~ConnectionId, 1366
  - cloneDataSet, 1366
  - COMPARATOR, 1366
  - compareTo, 1366
  - ConnectionId, 1366
  - copyDataSet, 1367
  - equals, 1367
  - getDataSetType, 1367
  - getValue, 1367, 1368
  - ID\_CONNECTIONID, 1368
  - operator<, 1368
  - operator=, 1368
  - operator==, 1368
  - setValue, 1368
  - toString, 1368
  - value, 1368
- activemq::commands::ConnectionInfo, 1393
  - ~ConnectionInfo, 1395
  - brokerMasterConnector, 1399
  - brokerPath, 1399
  - clientId, 1399
  - clientMaster, 1399
  - cloneDataSet, 1395
  - connectionId, 1399
  - ConnectionInfo, 1395
  - copyDataSet, 1395
  - createRemoveCommand, 1395
  - equals, 1395
  - faultTolerant, 1399
  - getBrokerPath, 1395, 1396
  - getClientId, 1396
  - getConnectionId, 1396
  - getDataSetType, 1396
  - getPassword, 1396, 1397
  - getUserName, 1397
  - ID\_CONNECTIONINFO, 1399
  - isBrokerMasterConnector, 1397
  - isClientMaster, 1397
  - isConnectionInfo, 1397
  - isFaultTolerant, 1397
  - isManageable, 1398
  - manageable, 1399
  - password, 1399
  - setBrokerMasterConnector, 1398
  - setBrokerPath, 1398
  - setClientId, 1398
  - setClientMaster, 1398
  - setConnectionId, 1398
  - setFaultTolerant, 1398
  - setManageable, 1398
  - setPassword, 1398
  - setUserName, 1398
  - toString, 1398
  - userName, 1399
  - visit, 1398
- activemq::commands::ConsumerControl, 1441
  - ~ConsumerControl, 1443
  - cloneDataSet, 1443
  - close, 1446
  - ConsumerControl, 1443

- consumerId, 1446
- copyDataStructure, 1443
- destination, 1446
- equals, 1443
- flush, 1446
- getConsumerId, 1443, 1444
- getDataStructureType, 1444
- getDestination, 1444, 1445
- getPrefetch, 1445
- ID\_CONSUMERCONTROL, 1446
- isClose, 1445
- isFlush, 1445
- isStart, 1445
- isStop, 1445
- prefetch, 1446
- setClose, 1445
- setConsumerId, 1445
- setDestination, 1445
- setFlush, 1445
- setPrefetch, 1445
- setStart, 1445
- setStop, 1445
- start, 1446
- stop, 1446
- toString, 1445
- visit, 1446
- activemq::commands::ConsumerId, 1472
  - ~ConsumerId, 1474
  - cloneDataStructure, 1474
  - COMPARATOR, 1474
  - compareTo, 1474
  - connectionId, 1476
  - ConsumerId, 1474
  - copyDataStructure, 1474
  - equals, 1474, 1475
  - getConnectionId, 1475
  - getDataStructureType, 1475
  - getParentId, 1475
  - getSessionId, 1476
  - getValue, 1476
  - ID\_CONSUMERID, 1476
  - operator<, 1476
  - operator=, 1476
  - operator==, 1476
  - sessionId, 1476
  - setConnectionId, 1476
  - setSessionId, 1476
  - setValue, 1476
  - toString, 1476
  - value, 1477
- activemq::commands::ConsumerInfo, 1501
  - ~ConsumerInfo, 1504
  - additionalPredicate, 1509
  - brokerPath, 1509
  - browser, 1509
  - cloneDataStructure, 1504
  - consumerId, 1509
  - ConsumerInfo, 1504
  - copyDataStructure, 1504
  - createRemoveCommand, 1504
  - destination, 1509
  - dispatchAsync, 1509
  - equals, 1504
  - exclusive, 1509
  - getAdditionalPredicate, 1504, 1505
  - getBrokerPath, 1505
  - getConsumerId, 1505
  - getDataStructureType, 1505
  - getDestination, 1505, 1506
  - getMaximumPendingMessageLimit, 1506
  - getNetworkConsumerPath, 1506
  - getPrefetchSize, 1506
  - getPriority, 1506
  - getSelector, 1506
  - getSubscriptionName, 1506
  - ID\_CONSUMERINFO, 1509
  - isBrowser, 1506
  - isConsumerInfo, 1506
  - isDispatchAsync, 1506
  - isExclusive, 1507
  - isNetworkSubscription, 1507
  - isNoLocal, 1507
  - isNoRangeAcks, 1507
  - isOptimizedAcknowledge, 1507
  - isRetroactive, 1507
  - maximumPendingMessageLimit, 1509
  - networkConsumerPath, 1509
  - networkSubscription, 1509
  - noLocal, 1509
  - noRangeAcks, 1509
  - optimizedAcknowledge, 1509
  - prefetchSize, 1509
  - priority, 1509
  - retroactive, 1509
  - selector, 1509
  - setAdditionalPredicate, 1507
  - setBrokerPath, 1507
  - setBrowser, 1507
  - setConsumerId, 1507
  - setDestination, 1507

- setDispatchAsync, 1507
- setExclusive, 1507
- setMaximumPendingMessageLimit, 1507
- setNetworkConsumerPath, 1507
- setNetworkSubscription, 1507
- setNoLocal, 1507
- setNoRangeAcks, 1507
- setOptimizedAcknowledge, 1507
- setPrefetchSize, 1507
- setPriority, 1507
- setRetroactive, 1507
- setSelector, 1507
- setSubscriptionName, 1507
- subscriptionName, 1509
- toString, 1507
- visit, 1508
- activemq::commands::ControlCommand, 1536
  - ~ControlCommand, 1537
  - cloneDataStructure, 1537
  - command, 1539
  - ControlCommand, 1537
  - copyDataStructure, 1537
  - equals, 1538
  - getCommand, 1538
  - getDataStructureType, 1538
  - ID\_CONTROLCOMMAND, 1539
  - setCommand, 1538
  - toString, 1538
  - visit, 1538
- activemq::commands::DataArrayResponse, 1572
  - ~DataArrayResponse, 1573
  - cloneDataStructure, 1573
  - copyDataStructure, 1573
  - data, 1574
  - DataArrayResponse, 1573
  - equals, 1573
  - getData, 1573, 1574
  - getDataStructureType, 1574
  - ID\_DATAARRAYRESPONSE, 1574
  - setData, 1574
  - toString, 1574
- activemq::commands::DataResponse, 1632
  - ~DataResponse, 1633
  - cloneDataStructure, 1633
  - copyDataStructure, 1633
  - data, 1634
  - DataResponse, 1633
  - equals, 1633
  - getData, 1633, 1634
  - getDataStructureType, 1634
  - ID\_DATARESPONSE, 1634
  - setData, 1634
  - toString, 1634
- activemq::commands::DestinationInfo, 1779
  - ~DestinationInfo, 1781
  - brokerPath, 1784
  - cloneDataStructure, 1781
  - connectionId, 1784
  - copyDataStructure, 1781
  - destination, 1784
  - DestinationInfo, 1781
  - equals, 1781
  - getBrokerPath, 1782
  - getConnectionId, 1782
  - getDataStructureType, 1782
  - getDestination, 1782, 1783
  - getOperationType, 1783
  - getTimeout, 1783
  - ID\_DESTINATIONINFO, 1784
  - operationType, 1784
  - setBrokerPath, 1783
  - setConnectionId, 1783
  - setDestination, 1783
  - setOperationType, 1783
  - setTimeout, 1783
  - timeout, 1784
  - toString, 1783
  - visit, 1783
- activemq::commands::DiscoveryEvent, 1812
  - ~DiscoveryEvent, 1813
  - brokerName, 1815
  - cloneDataStructure, 1813
  - copyDataStructure, 1813
  - DiscoveryEvent, 1813
  - equals, 1813
  - getBrokerName, 1814
  - getDataStructureType, 1814
  - getServiceName, 1814
  - ID\_DISCOVERYEVENT, 1815
  - serviceName, 1815
  - setBrokerName, 1814
  - setServiceName, 1814



- toString, 1814
- activemq::commands::ExceptionResponse, 1894
  - ~ExceptionResponse, 1896
  - cloneDataStructure, 1896
  - copyDataStructure, 1896
  - equals, 1896
  - exception, 1897
  - ExceptionResponse, 1896
  - getDataStructureType, 1896
  - getException, 1897
  - ID\_EXCEPTIONRESPONSE, 1897
  - setException, 1897
  - toString, 1897
- activemq::commands::FlushCommand, 1999
  - ~FlushCommand, 2000
  - cloneDataStructure, 2000
  - copyDataStructure, 2000
  - equals, 2000
  - FlushCommand, 2000
  - getDataStructureType, 2000
  - ID\_FLUSHCOMMAND, 2001
  - toString, 2001
  - visit, 2001
- activemq::commands::IntegerResponse, 2160
  - ~IntegerResponse, 2161
  - cloneDataStructure, 2161
  - copyDataStructure, 2161
  - equals, 2161
  - getDataStructureType, 2161
  - getResult, 2162
  - ID\_INTEGERRESPONSE, 2162
  - IntegerResponse, 2161
  - result, 2162
  - setResult, 2162
  - toString, 2162
- activemq::commands::JournalQueueAck, 2224
  - ~JournalQueueAck, 2225
  - cloneDataStructure, 2225
  - copyDataStructure, 2225
  - destination, 2227
  - equals, 2226
  - getDataStructureType, 2226
  - getDestination, 2226, 2227
  - getMessageAck, 2227
  - ID\_JOURNALQUEUEACK, 2227
  - JournalQueueAck, 2225
  - messageAck, 2227
  - setDestination, 2227
  - setMessageAck, 2227
- activemq::commands::JournalTopicAck, 2252
  - ~JournalTopicAck, 2253
  - clientId, 2256
  - cloneDataStructure, 2253
  - copyDataStructure, 2253
  - destination, 2256
  - equals, 2254
  - getClientId, 2254
  - getDataStructureType, 2254
  - getDestination, 2254, 2255
  - getMessageId, 2255
  - getMessageSequenceId, 2255
  - getSubscriptionName, 2255
  - getTransactionId, 2255
  - ID\_JOURNALTOPICACK, 2256
  - JournalTopicAck, 2253
  - messageId, 2256
  - messageSequenceId, 2256
  - setClientId, 2255
  - setDestination, 2255
  - setMessageId, 2255
  - setMessageSequenceId, 2255
  - setSubscriptionName, 2255
  - setTransactionId, 2255
  - subscriptionName, 2256
  - toString, 2255
  - transactionId, 2256
- activemq::commands::JournalTrace, 2281
  - ~JournalTrace, 2282
  - cloneDataStructure, 2282
  - copyDataStructure, 2282
  - equals, 2282
  - getDataStructureType, 2282
  - getMessage, 2283
  - ID\_JOURNALTRACE, 2283
  - JournalTrace, 2282
  - message, 2283
  - setMessage, 2283
  - toString, 2283
- activemq::commands::JournalTransaction, 2308
  - ~JournalTransaction, 2309
  - cloneDataStructure, 2309
  - copyDataStructure, 2309
  - equals, 2309
  - getDataStructureType, 2310
  - getTransactionId, 2310
  - getType, 2310
  - getWasPrepared, 2310

- ID\_JOURNALTRANSACTION, 2311
- JournalTransaction, 2309
- setTransactionId, 2310
- setType, 2310
- setWasPrepared, 2310
- toString, 2310
- transactionId, 2311
- type, 2311
- wasPrepared, 2311
- activemq::commands::KeepAliveInfo, 2335
  - ~KeepAliveInfo, 2336
  - cloneDataStructure, 2336
  - copyDataStructure, 2336
  - equals, 2337
  - getDataStructureType, 2337
  - ID\_KEEPLIVEINFO, 2338
  - isKeepAliveInfo, 2337
  - KeepAliveInfo, 2336
  - toString, 2337
  - visit, 2337
- activemq::commands::LastPartialCommand, 2371
  - ~LastPartialCommand, 2372
  - cloneDataStructure, 2372
  - copyDataStructure, 2372
  - equals, 2372
  - getDataStructureType, 2373
  - ID\_LASTPARTIALCOMMAND, 2373
  - LastPartialCommand, 2372
  - toString, 2373
- activemq::commands::LocalTransactionId, 2420
  - ~LocalTransactionId, 2422
  - cloneDataStructure, 2422
  - COMPARATOR, 2421
  - compareTo, 2422
  - connectionId, 2424
  - copyDataStructure, 2422
  - equals, 2422, 2423
  - getConnectionId, 2423
  - getDataStructureType, 2423
  - getValue, 2423
  - ID\_LOCALTRANSACTIONID, 2424
  - LocalTransactionId, 2422
  - operator<, 2424
  - operator=, 2424
  - operator==, 2424
  - setConnectionId, 2424
  - setValue, 2424
  - toString, 2424
  - value, 2424
- activemq::commands::Message, 2596
  - ~Message, 2601
  - afterUnmarshal, 2601
  - arrival, 2613
  - beforeMarshal, 2601
  - brokerInTime, 2613
  - brokerOutTime, 2613
  - brokerPath, 2613
  - cloneDataStructure, 2601
  - cluster, 2613
  - compressed, 2613
  - connection, 2613
  - content, 2613
  - copyDataStructure, 2602
  - correlationId, 2613
  - dataStructure, 2613
  - DEFAULT\_MESSAGE\_SIZE, 2613
  - destination, 2613
  - droppable, 2613
  - equals, 2602
  - expiration, 2613
  - getAckHandler, 2602
  - getArrival, 2603
  - getBrokerInTime, 2603
  - getBrokerOutTime, 2603
  - getBrokerPath, 2603
  - getCluster, 2603
  - getConnection, 2603
  - getContent, 2603, 2604
  - getCorrelationId, 2604
  - getDataStructure, 2604
  - getDataStructureType, 2604
  - getDestination, 2604, 2605
  - getExpiration, 2605
  - getGroupID, 2605
  - getGroupSequence, 2605
  - getMarshaledProperties, 2605
  - getMessageId, 2605
  - getMessageProperties, 2605
  - getOriginalDestination, 2606
  - getOriginalTransactionId, 2606
  - getPriority, 2606
  - getProducerId, 2606
  - getRedeliveryCounter, 2606
  - getReplyTo, 2606
  - getSize, 2606
  - getTargetConsumerId, 2606, 2607
  - getTimestamp, 2607
  - getTransactionId, 2607

getType, 2607  
getUserID, 2607  
groupID, 2613  
groupSequence, 2613  
ID\_MESSAGE, 2613  
isCompressed, 2607  
isDroppable, 2607  
isExpired, 2607  
isMarshalAware, 2607  
isMessage, 2608  
isPersistent, 2608  
isReadOnlyBody, 2608  
isReadOnlyProperties, 2608  
isRecievedByDFBridge, 2608  
marshalledProperties, 2613  
Message, 2601  
messageId, 2613  
onSend, 2608  
originalDestination, 2613  
originalTransactionId, 2613  
persistent, 2613  
priority, 2613  
producerId, 2613  
recievedByDFBridge, 2613  
redeliveryCounter, 2613  
replyTo, 2613  
setAckHandler, 2608  
setArrival, 2609  
setBrokerInTime, 2609  
setBrokerOutTime, 2609  
setBrokerPath, 2609  
setCluster, 2609  
setCompressed, 2609  
setConnection, 2609  
setContent, 2609  
setCorrelationId, 2610  
setDataStructure, 2610  
setDestination, 2610  
setDroppable, 2610  
setExpiration, 2610  
setGroupID, 2610  
setGroupSequence, 2610  
setMarshalledProperties, 2610  
setMessageId, 2610  
setOriginalDestination, 2610  
setOriginalTransactionId, 2610  
setPersistent, 2610  
setPriority, 2610  
setProducerId, 2610  
setReadOnlyBody, 2610  
setReadOnlyProperties, 2611  
setRecievedByDFBridge, 2611  
setRedeliveryCounter, 2611  
setReplyTo, 2611  
setTargetConsumerId, 2611  
setTimestamp, 2611  
setTransactionId, 2611  
setType, 2611  
setUserID, 2611  
targetConsumerId, 2613  
timestamp, 2613  
toString, 2611  
transactionId, 2613  
type, 2613  
userId, 2613  
visit, 2612  
activemq::commands::MessageAck, 2643  
  ~MessageAck, 2644  
  ackType, 2648  
  cloneDataStructure, 2644  
  consumerId, 2648  
  copyDataStructure, 2644  
  destination, 2648  
  equals, 2645  
  firstMessageId, 2648  
  getAckType, 2645  
  getConsumerId, 2645  
  getDataStructureType, 2645  
  getDestination, 2645, 2646  
  getFirstMessageId, 2646  
  getLastMessageId, 2646  
  getMessageCount, 2646  
  getTransactionId, 2646  
  ID\_MESSAGEACK, 2648  
  isMessageAck, 2646  
  lastMessageId, 2648  
  MessageAck, 2644  
  messageCount, 2648  
  setAckType, 2646  
  setConsumerId, 2647  
  setDestination, 2647  
  setFirstMessageId, 2647  
  setLastMessageId, 2647  
  setMessageCount, 2647  
  setTransactionId, 2647  
  toString, 2647  
  transactionId, 2648  
  visit, 2647  
activemq::commands::MessageDispatch, 2678  
  ~MessageDispatch, 2680

- cloneDataStructure, 2680
- consumerId, 2683
- copyDataStructure, 2680
- destination, 2683
- equals, 2680
- getConsumerId, 2681
- getDataStructureType, 2681
- getDestination, 2681
- getMessage, 2681
- getRedeliveryCounter, 2681
- ID\_MESSAGEDISPATCH, 2683
- isMessageDispatch, 2681
- message, 2683
- MessageDispatch, 2680
- redeliveryCounter, 2683
- setConsumerId, 2681
- setDestination, 2682
- setMessage, 2682
- setRedeliveryCounter, 2682
- toString, 2682
- visit, 2682
- activemq::commands::MessageDispatchNotification, 2716
  - ~MessageDispatchNotification, 2717
  - cloneDataStructure, 2717
  - consumerId, 2720
  - copyDataStructure, 2717
  - deliverySequenceId, 2720
  - destination, 2720
  - equals, 2718
  - getConsumerId, 2718
  - getDataStructureType, 2718
  - getDeliverySequenceId, 2718
  - getDestination, 2719
  - getMessageId, 2719
  - ID\_MESSAGEDISPATCHNOTIFICATION, 2720
  - isMessageDispatchNotification, 2719
  - MessageDispatchNotification, 2717
  - messageId, 2720
  - setConsumerId, 2719
  - setDeliverySequenceId, 2719
  - setDestination, 2719
  - setMessageId, 2719
  - toString, 2719
  - visit, 2720
- activemq::commands::MessageId, 2750
  - ~MessageId, 2752
  - brokerSequenceId, 2755
  - cloneDataStructure, 2752
  - COMPARATOR, 2752
  - compareTo, 2752
  - copyDataStructure, 2752
  - equals, 2753
  - getBrokerSequenceId, 2753
  - getDataStructureType, 2753
  - getProducerId, 2753, 2754
  - getProducerSequenceId, 2754
  - ID\_MESSAGEID, 2755
  - MessageId, 2752
  - operator<, 2754
  - operator=, 2754
  - operator==, 2754
  - producerId, 2755
  - producerSequenceId, 2755
  - setBrokerSequenceId, 2754
  - setProducerId, 2754
  - setProducerSequenceId, 2754
  - setTextView, 2754
  - setValue, 2754
  - toString, 2754
- activemq::commands::MessagePull, 2824
  - ~MessagePull, 2825
  - cloneDataStructure, 2825
  - consumerId, 2828
  - copyDataStructure, 2825
  - correlationId, 2828
  - destination, 2828
  - equals, 2826
  - getConsumerId, 2826
  - getCorrelationId, 2826
  - getDataStructureType, 2826
  - getDestination, 2826, 2827
  - getMessageId, 2827
  - getTimeout, 2827
  - ID\_MESSAGEPULL, 2828
  - messageId, 2828
  - MessagePull, 2825
  - setConsumerId, 2827
  - setCorrelationId, 2827
  - setDestination, 2827
  - setMessageId, 2827
  - setTimeout, 2827
  - timeout, 2828
  - toString, 2827
  - visit, 2827
- activemq::commands::NetworkBridgeFilter, 2877
  - ~NetworkBridgeFilter, 2878
  - cloneDataStructure, 2878

- copyDataStructure, 2878
- equals, 2879
- getDataStructureType, 2879
- getNetworkBrokerId, 2879, 2880
- getNetworkTTL, 2880
- ID\_NETWORKBRIDGEFILTER, 2880
- NetworkBridgeFilter, 2878
- networkBrokerId, 2880
- networkTTL, 2880
- setNetworkBrokerId, 2880
- setNetworkTTL, 2880
- toString, 2880
- activemq::commands::PartialCommand, 3002
  - ~PartialCommand, 3003
  - cloneDataStructure, 3003
  - commandId, 3005
  - copyDataStructure, 3003
  - data, 3005
  - equals, 3003
  - getCommandId, 3004
  - getData, 3004
  - getDataStructureType, 3004
  - ID\_PARTIALCOMMAND, 3005
  - PartialCommand, 3003
  - setCommandId, 3004
  - setData, 3005
  - toString, 3005
- activemq::commands::ProducerAck, 3124
  - ~ProducerAck, 3126
  - cloneDataStructure, 3126
  - copyDataStructure, 3126
  - equals, 3126
  - getDataStructureType, 3126
  - getProducerId, 3127
  - getSize, 3127
  - ID\_PRODUCERACK, 3128
  - isProducerAck, 3127
  - ProducerAck, 3126
  - producerId, 3128
  - setProducerId, 3127
  - setSize, 3127
  - size, 3128
  - toString, 3127
  - visit, 3127
- activemq::commands::ProducerId, 3157
  - ~ProducerId, 3158
  - cloneDataStructure, 3158
  - COMPARATOR, 3158
  - compareTo, 3159
  - connectionId, 3161
  - copyDataStructure, 3159
  - equals, 3159
  - getConnectionId, 3159
  - getDataStructureType, 3159
  - getParentId, 3160
  - getSessionId, 3160
  - getValue, 3160
  - ID\_PRODUCERID, 3161
  - operator<, 3160
  - operator=, 3160
  - operator==, 3160
  - ProducerId, 3158
  - sessionId, 3161
  - setConnectionId, 3160
  - setProducerSessionKey, 3160
  - setSessionId, 3160
  - setValue, 3160
  - toString, 3160
  - value, 3161
- activemq::commands::ProducerInfo, 3185
  - ~ProducerInfo, 3187
  - brokerPath, 3190
  - cloneDataStructure, 3187
  - copyDataStructure, 3187
  - createRemoveCommand, 3187
  - destination, 3190
  - dispatchAsync, 3190
  - equals, 3187
  - getBrokerPath, 3187, 3188
  - getDataStructureType, 3188
  - getDestination, 3188
  - getProducerId, 3188
  - getWindowSize, 3188
  - ID\_PRODUCERINFO, 3190
  - isDispatchAsync, 3188
  - isProducerInfo, 3188
  - producerId, 3190
  - ProducerInfo, 3187
  - setBrokerPath, 3188
  - setDestination, 3189
  - setDispatchAsync, 3189
  - setProducerId, 3189
  - setWindowSize, 3189
  - toString, 3189
  - visit, 3189
  - windowSize, 3190
- activemq::commands::RemoveInfo, 3284
  - ~RemoveInfo, 3285
  - cloneDataStructure, 3285
  - copyDataStructure, 3285

- equals, 3286
  - getDataStructureType, 3286
  - getLastDeliveredSequenceId, 3286
  - getObjectId, 3286
  - ID\_REMOVEINFO, 3287
  - isRemoveInfo, 3286
  - lastDeliveredSequenceId, 3287
  - objectId, 3287
  - RemoveInfo, 3285
  - setLastDeliveredSequenceId, 3286
  - setObjectId, 3287
  - toString, 3287
  - visit, 3287
- activemq::commands::RemoveSubscriptionInfo, 3313
  - ~RemoveSubscriptionInfo, 3315
  - clientId, 3318
  - cloneDataStructure, 3315
  - connectionId, 3318
  - copyDataStructure, 3315
  - equals, 3315
  - getClientId, 3315, 3316
  - getConnectionId, 3316
  - getDataStructureType, 3316
  - getSubscriptionName, 3316
  - ID\_REMOVESUBSCRIPTIONINFO, 3318
  - isRemoveSubscriptionInfo, 3316
  - RemoveSubscriptionInfo, 3315
  - setClientId, 3316
  - setConnectionId, 3317
  - setSubscriptionName, 3317
  - subscriptionName, 3318
  - toString, 3317
  - visit, 3317
- activemq::commands::ReplayCommand, 3343
  - ~ReplayCommand, 3345
  - cloneDataStructure, 3345
  - copyDataStructure, 3345
  - equals, 3345
  - firstNakNumber, 3347
  - getDataStructureType, 3345
  - getFirstNakNumber, 3346
  - getLastNakNumber, 3346
  - ID\_REPLAYCOMMAND, 3347
  - lastNakNumber, 3347
  - ReplayCommand, 3345
  - setFirstNakNumber, 3346
  - setLastNakNumber, 3346
  - toString, 3346
- visit, 3346
- activemq::commands::Response, 3379
  - ~Response, 3380
  - cloneDataStructure, 3380
  - copyDataStructure, 3380
  - correlationId, 3382
  - equals, 3381
  - getCorrelationId, 3381
  - getDataStructureType, 3381
  - ID\_RESPONSE, 3382
  - isResponse, 3381
  - Response, 3380
  - setCorrelationId, 3381
  - toString, 3382
  - visit, 3382
- activemq::commands::SessionId, 3476
  - ~SessionId, 3478
  - cloneDataStructure, 3478
  - COMPARATOR, 3478
  - compareTo, 3478
  - connectionId, 3480
  - copyDataStructure, 3478
  - equals, 3478, 3479
  - getConnectionId, 3479
  - getDataStructureType, 3479
  - getParentId, 3479
  - getValue, 3480
  - ID\_SESSIONID, 3480
  - operator<, 3480
  - operator=, 3480
  - operator==, 3480
  - SessionId, 3478
  - setConnectionId, 3480
  - setValue, 3480
  - toString, 3480
  - value, 3480
- activemq::commands::SessionInfo, 3505
  - ~SessionInfo, 3506
  - cloneDataStructure, 3506
  - copyDataStructure, 3506
  - createRemoveCommand, 3506
  - equals, 3507
  - getAckMode, 3507
  - getDataStructureType, 3507
  - getSessionId, 3507
  - ID\_SESSIONINFO, 3508
  - sessionId, 3508
  - SessionInfo, 3506
  - setAckMode, 3507
  - setSessionId, 3507

- toString, 3507
- visit, 3508
- activemq::commands::ShutdownInfo, 3573
  - ~ShutdownInfo, 3574
  - cloneDataStructure, 3574
  - copyDataStructure, 3574
  - equals, 3574
  - getDataStructureType, 3574
  - ID\_SHUTDOWNINFO, 3576
  - isShutdownInfo, 3575
  - ShutdownInfo, 3574
  - toString, 3575
  - visit, 3575
- activemq::commands::SubscriptionInfo, 3782
  - ~SubscriptionInfo, 3783
  - clientId, 3786
  - cloneDataStructure, 3783
  - copyDataStructure, 3783
  - destination, 3786
  - equals, 3784
  - getClientId, 3784
  - getDataStructureType, 3784
  - getDestination, 3784, 3785
  - getSelector, 3785
  - getSubscriptionName, 3785
  - getSubscribedDestination, 3785
  - ID\_SUBSCRIPTIONINFO, 3786
  - selector, 3786
  - setClientId, 3785
  - setDestination, 3785
  - setSelector, 3785
  - setSubscriptionName, 3785
  - setSubscribedDestination, 3785
  - subscriptionName, 3786
  - subscribedDestination, 3786
  - SubscriptionInfo, 3783
  - toString, 3785
- activemq::commands::TransactionId, 3932
  - ~TransactionId, 3933
  - cloneDataStructure, 3933
  - COMPARATOR, 3933
  - compareTo, 3934
  - copyDataStructure, 3934
  - equals, 3934
  - getDataStructureType, 3934
  - ID\_TRANSACTIONID, 3935
  - operator<, 3935
  - operator=, 3935
  - operator==, 3935
  - toString, 3935
  - TransactionId, 3933
- activemq::commands::TransactionInfo, 3959
  - ~TransactionInfo, 3961
  - cloneDataStructure, 3961
  - connectionId, 3964
  - copyDataStructure, 3961
  - equals, 3961
  - getConnectionId, 3961, 3962
  - getDataStructureType, 3962
  - getTransactionId, 3962
  - getType, 3962
  - ID\_TRANSACTIONINFO, 3964
  - isTransactionInfo, 3962
  - setConnectionId, 3962
  - setTransactionId, 3963
  - setType, 3963
  - toString, 3963
  - transactionId, 3964
  - TransactionInfo, 3961
  - type, 3964
  - visit, 3963
- activemq::commands::WireFormatInfo, 4093
  - ~WireFormatInfo, 4096
  - afterUnmarshal, 4096
  - beforeMarshal, 4096
  - cloneDataStructure, 4097
  - copyDataStructure, 4097
  - equals, 4097
  - getCacheSize, 4097
  - getDataStructureType, 4098
  - getMagic, 4098
  - getMarshaledProperties, 4098
  - getMaxInactivityDuration, 4098
  - getMaxInactivityDurationInitialDelay, 4098
  - getProperties, 4099
  - getVersion, 4099
  - ID\_WIREFORMATINFO, 4104
  - isCacheEnabled, 4099
  - isMarshalAware, 4099
  - isSizePrefixDisabled, 4100
  - isStackTraceEnabled, 4100
  - isTcpNoDelayEnabled, 4100
  - isTightEncodingEnabled, 4100
  - isValid, 4100
  - isWireFormatInfo, 4101
  - setCacheEnabled, 4101
  - setCacheSize, 4101
  - setMagic, 4101
  - setMarshaledProperties, 4101

- setMaxInactivityDuration, 4102
- setMaxInactivityDurationInitialDelay, 4102
- setProperties, 4102
- setSizePrefixDisabled, 4102
- setStackTraceEnabled, 4103
- setTcpNoDelayEnabled, 4103
- setTightEncodingEnabled, 4103
- setVersion, 4103
- toString, 4103
- visit, 4104
- WireFormatInfo, 4096
- activemq::commands::XATransactionId, 4143
  - ~XATransactionId, 4144
  - branchQualifier, 4147
  - cloneDataStructure, 4144
  - COMPARATOR, 4144
  - compareTo, 4145
  - copyDataStructure, 4145
  - equals, 4145
  - formatId, 4147
  - getBranchQualifier, 4145
  - getDataStructureType, 4145
  - getFormatId, 4146
  - getGlobalTransactionId, 4146
  - globalTransactionId, 4147
  - ID\_XATRANSACTIONID, 4147
  - operator<, 4146
  - operator=, 4146
  - operator==, 4146
  - setBranchQualifier, 4146
  - setFormatId, 4146
  - setGlobalTransactionId, 4146
  - toString, 4146
  - XATransactionId, 4144
- activemq::core, 96
- activemq::core::ActiveMQAckHandler, 183
  - ~ActiveMQAckHandler, 184
  - acknowledgeMessage, 184
- activemq::core::ActiveMQConnection, 260
  - ~ActiveMQConnection, 266
  - ActiveMQConnection, 266
  - addDispatcher, 266
  - addProducer, 266
  - addTransportListener, 267
  - close, 267
  - createSession, 267, 268
  - destroyDestination, 268
  - fire, 269
  - getBrokerURL, 269
  - getClientID, 269
  - getCloseTimeout, 270
  - getConnectionId, 270
  - getConnectionInfo, 270
  - getExceptionListener, 270
  - getMetaData, 271
  - getNextLocalTransactionId, 271
  - getNextSessionId, 271
  - getNextTempDestinationId, 271
  - getPassword, 271
  - getPrefetchPolicy, 272
  - getProducerWindowSize, 272
  - getRedeliveryPolicy, 272
  - getSendTimeout, 272
  - getTransport, 272
  - getUsername, 273
  - isAlwaysSyncSend, 273
  - isClosed, 273
  - isDispatchAsync, 273
  - isStarted, 273
  - isTransportFailed, 273
  - isUseAsyncSend, 274
  - isUseCompression, 274
  - onCommand, 274
  - oneway, 274
  - onException, 275
  - removeDispatcher, 275
  - removeProducer, 275
  - removeSession, 275
  - removeTransportListener, 275
  - sendPullRequest, 276
  - setAlwaysSyncSend, 276
  - setBrokerURL, 276
  - setClientID, 276
  - setCloseTimeout, 277
  - setDefaultClientID, 277
  - setDispatchAsync, 277
  - setExceptionListener, 278
  - setPassword, 278
  - setPrefetchPolicy, 278
  - setProducerWindowSize, 278
  - setRedeliveryPolicy, 279
  - setSendTimeout, 279
  - setTransportInterruptProcessingComplete, 279
  - setUseAsyncSend, 279
  - setUseCompression, 280
  - setUsername, 280
  - start, 280
  - stop, 280



- syncRequest, 280
- transportInterrupted, 281
- transportResumed, 281
- activemq::core::ActiveMQConnectionFactory, 281
  - ~ActiveMQConnectionFactory, 284
  - ActiveMQConnectionFactory, 284
  - createConnection, 285, 286
  - DEFAULT\_URI, 293
  - getBrokerURL, 286
  - getClientId, 287
  - getCloseTimeout, 287
  - getExceptionListener, 287
  - getPassword, 287
  - getPrefetchPolicy, 287
  - getProducerWindowSize, 288
  - getRedeliveryPolicy, 288
  - getSendTimeout, 288
  - getUsername, 288
  - isAlwaysSyncSend, 289
  - isDispatchAsync, 289
  - isUseAsyncSend, 289
  - isUseCompression, 289
  - setAlwaysSyncSend, 289
  - setBrokerURL, 289
  - setClientId, 290
  - setCloseTimeout, 290
  - setDispatchAsync, 290
  - setExceptionListener, 290
  - setPassword, 291
  - setPrefetchPolicy, 291
  - setProducerWindowSize, 291
  - setRedeliveryPolicy, 291
  - setSendTimeout, 292
  - setUseAsyncSend, 292
  - setUseCompression, 292
  - setUsername, 292
- activemq::core::ActiveMQConnectionMetaData, 293
  - ~ActiveMQConnectionMetaData, 294
  - ActiveMQConnectionMetaData, 294
  - getCMSMajorVersion, 294
  - getCMSMinorVersion, 294
  - getCMSProviderName, 295
  - getCMSVersion, 295
  - getCMSXPropertyNames, 295
  - getProviderMajorVersion, 296
  - getProviderMinorVersion, 296
  - getProviderVersion, 296
- activemq::core::ActiveMQConstants, 297
  - ACK\_TYPE\_CONSUMED, 298
  - ACK\_TYPE\_DELIVERED, 298
  - ACK\_TYPE\_INDIVIDUAL, 298
  - ACK\_TYPE\_POISON, 298
  - ACK\_TYPE\_REDELIVERED, 298
  - AckType, 298
  - CONNECTION\_ALWAYS\_SYNC\_SEND, 299
  - CONNECTION\_CLOSE\_TIMEOUT, 299
  - CONNECTION\_DISPATCH\_ASYNC, 300
  - CONNECTION\_PRODUCER\_WINDOW\_SIZE, 299
  - CONNECTION\_SEND\_TIMEOUT, 299
  - CONNECTION\_USE\_ASYNC\_SEND, 299
  - CONNECTION\_USE\_COMPRESSION, 299
  - CONSUMER\_DISPATCH\_ASYNC, 299
  - CONSUMER\_EXCLUSIVE, 299
  - CONSUMER\_NOLOCAL, 299
  - CONSUMER\_PREFETCH\_SIZE, 299
  - CONSUMER\_PRIORITY, 299
  - CONSUMER\_RETROACTIVE, 299
  - CONSUMER\_SELECTOR, 299
  - CUNSUMER\_MAX\_PENDING\_MSG\_LIMIT, 299
  - DESTINATION\_ADD\_OPERATION, 298
  - DESTINATION\_REMOVE\_OPERATION, 298
  - DestinationActions, 298
  - DestinationOption, 298
  - NUM\_OPTIONS, 299
  - NUM\_PARAMS, 300
  - PARAM\_CLIENTID, 300
  - PARAM\_PASSWORD, 300
  - PARAM\_USERNAME, 300
  - toDestinationOption, 300
  - toString, 300
  - toURIOption, 300
  - TRANSACTION\_STATE\_BEGIN, 299
  - TRANSACTION\_STATE\_COMMIT\_ONE\_PHASE, 299
  - TRANSACTION\_STATE\_COMMIT\_TWO\_PHASE, 299
  - TRANSACTION\_STATE\_END, 299
  - TRANSACTION\_STATE\_FORGET, 299

- TRANSACTION\_STATE\_PREPARE, 299
- TRANSACTION\_STATE\_RECOVER, 299
- TRANSACTION\_STATE\_ROLLBACK, 299
- TransactionState, 299
- URIParam, 299
- activemq::core::ActiveMQConstants::StaticInitializer, 3693
  - ~StaticInitializer, 3693
  - destOptionMap, 3693
  - destOptions, 3693
  - StaticInitializer, 3693
  - uriParams, 3693
  - uriParamsMap, 3693
- activemq::core::ActiveMQConsumer, 300
  - ~ActiveMQConsumer, 303
  - acknowledge, 303
  - ActiveMQConsumer, 303
  - afterMessageIsConsumed, 304
  - beforeMessageIsConsumed, 304
  - clearMessagesInProgress, 304
  - close, 304
  - commit, 305
  - deliverAcks, 305
  - dequeue, 305
  - dispatch, 305
  - doClose, 305
  - getConsumerId, 306
  - getConsumerInfo, 306
  - getLastDeliveredSequenceId, 306
  - getMessageAvailableCount, 306
  - getMessageListener, 306
  - getMessageSelector, 307
  - getRedeliveryPolicy, 307
  - inProgressClearRequired, 307
  - isClosed, 307
  - isSynchronizationRegistered, 307
  - iterate, 308
  - receive, 308
  - receiveNoWait, 309
  - rollback, 309
  - setLastDeliveredSequenceId, 309
  - setMessageListener, 309
  - setRedeliveryPolicy, 310
  - setSynchronizationRegistered, 310
  - start, 310
  - stop, 310
- activemq::core::ActiveMQProducer, 466
  - ~ActiveMQProducer, 468
  - ActiveMQProducer, 468
  - close, 468
  - getDeliveryMode, 468
  - getDisableMessageID, 469
  - getDisableMessageTimeStamp, 469
  - getPriority, 469
  - getProducerId, 469
  - getProducerInfo, 470
  - getSendTimeout, 470
  - getTimeToLive, 470
  - isClosed, 470
  - onProducerAck, 470
  - send, 471, 472
  - setDeliveryMode, 473
  - setDisableMessageID, 473
  - setDisableMessageTimeStamp, 473
  - setPriority, 473
  - setSendTimeout, 474
  - setTimeToLive, 474
- activemq::core::ActiveMQQueueBrowser, 483
  - ~ActiveMQQueueBrowser, 484
  - ActiveMQQueueBrowser, 484
  - Browser, 486
  - close, 484
  - getEnumeration, 485
  - getMessageSelector, 485
  - getQueue, 485
  - hasMoreMessages, 486
  - nextMessage, 486
- activemq::core::ActiveMQSession, 512
  - ~ActiveMQSession, 517
  - acknowledge, 517
  - ActiveMQSession, 517
  - ActiveMQSessionExecutor, 532
  - addConsumer, 517
  - addProducer, 517
  - clearMessagesInProgress, 518
  - close, 518
  - commit, 518
  - createBrowser, 518, 519
  - createBytesMessage, 519
  - createConsumer, 520, 521
  - createDurableConsumer, 521
  - createMapMessage, 521
  - createMessage, 522
  - createProducer, 522
  - createQueue, 522
  - createStreamMessage, 523

- createTemporaryQueue, 523
- createTemporaryTopic, 523
- createTextMessage, 523, 524
- createTopic, 524
- deliverAcks, 524
- dispatch, 525
- doStartTransaction, 525
- fire, 525
- getAcknowledgeMode, 525
- getConnection, 525
- getExceptionListener, 526
- getLastDeliveredSequenceId, 526
- getNextConsumerId, 526
- getNextProducerId, 526
- getSessionId, 526
- getSessionInfo, 527
- getTransactionContext, 527
- isAutoAcknowledge, 527
- isClientAcknowledge, 527
- isDupsOkAcknowledge, 527
- isIndividualAcknowledge, 527
- isStarted, 528
- isTransacted, 528
- oneway, 528
- recover, 528
- redispach, 529
- removeConsumer, 529
- removeProducer, 529
- rollback, 530
- send, 530
- setLastDeliveredSequenceId, 530
- start, 531
- stop, 531
- syncRequest, 531
- unsubscribe, 531
- wakeup, 531
- activemq::core::ActiveMQSessionExecutor, 532
  - ~ActiveMQSessionExecutor, 533
  - ActiveMQSessionExecutor, 533
  - clear, 533
  - clearMessagesInProgress, 533
  - close, 533
  - execute, 534
  - executeFirst, 534
  - getUnconsumedMessages, 534
  - hasUnconsumedMessages, 534
  - isEmpty, 534
  - isRunning, 535
  - iterate, 535
  - start, 535
  - stop, 535
  - wakeup, 535
- activemq::core::ActiveMQTransactionContext, 727
  - ~ActiveMQTransactionContext, 728
  - ActiveMQTransactionContext, 728
  - addSynchronization, 728
  - begin, 728
  - commit, 728
  - getTransactionId, 729
  - isInTransaction, 729
  - removeSynchronization, 729
  - rollback, 729
- activemq::core::DispatchData, 1839
  - DispatchData, 1840
  - getConsumerId, 1840
  - getMessage, 1840
- activemq::core::Dispatcher, 1840
  - ~Dispatcher, 1841
  - dispatch, 1841
- activemq::core::MessageDispatchChannel, 2683
  - ~MessageDispatchChannel, 2685
  - clear, 2685
  - close, 2685
  - dequeue, 2685
  - dequeueNoWait, 2685
  - enqueue, 2686
  - enqueueFirst, 2686
  - isClosed, 2686
  - isEmpty, 2686
  - isRunning, 2686
  - lock, 2687
  - MessageDispatchChannel, 2685
  - notify, 2687
  - notifyAll, 2687
  - peek, 2687
  - removeAll, 2688
  - size, 2688
  - start, 2688
  - stop, 2688
  - tryLock, 2688
  - unlock, 2689
  - wait, 2689, 2690
- activemq::core::policies, 98
- activemq::core::policies::DefaultPrefetchPolicy, 1726
  - ~DefaultPrefetchPolicy, 1727
  - clone, 1727

- DEFAULT\_DURABLE\_TOPIC\_PREFETCH, 1730
- DEFAULT\_QUEUE\_BROWSER\_PREFETCH, 1730
- DEFAULT\_QUEUE\_PREFETCH, 1730
- DEFAULT\_TOPIC\_PREFETCH, 1730
- DefaultPrefetchPolicy, 1727
- getDurableTopicPrefetch, 1728
- getMaxPrefetchLimit, 1728
- getQueueBrowserPrefetch, 1728
- getQueuePrefetch, 1728
- getTopicPrefetch, 1728
- MAX\_PREFETCH\_SIZE, 1730
- setDurableTopicPrefetch, 1729
- setQueueBrowserPrefetch, 1729
- setQueuePrefetch, 1729
- setTopicPrefetch, 1729
- activemq::core::policies::DefaultRedeliveryPolicy, 1730
  - ~DefaultRedeliveryPolicy, 1731
  - clone, 1731
  - DefaultRedeliveryPolicy, 1731
  - getBackOffMultiplier, 1731
  - getCollisionAvoidancePercent, 1732
  - getInitialRedeliveryDelay, 1732
  - getMaximumRedeliveries, 1732
  - getRedeliveryDelay, 1732
  - isUseCollisionAvoidance, 1733
  - isUseExponentialBackOff, 1733
  - setBackOffMultiplier, 1733
  - setCollisionAvoidancePercent, 1733
  - setInitialRedeliveryDelay, 1733
  - setMaximumRedeliveries, 1734
  - setUseCollisionAvoidance, 1734
  - setUseExponentialBackOff, 1734
- activemq::core::PrefetchPolicy, 3062
  - ~PrefetchPolicy, 3064
  - clone, 3064
  - configure, 3064
  - getDurableTopicPrefetch, 3065
  - getMaxPrefetchLimit, 3065
  - getQueueBrowserPrefetch, 3065
  - getQueuePrefetch, 3065
  - getTopicPrefetch, 3065
  - PrefetchPolicy, 3064
  - setDurableTopicPrefetch, 3066
  - setQueueBrowserPrefetch, 3066
  - setQueuePrefetch, 3066
  - setTopicPrefetch, 3066
- activemq::core::RedeliveryPolicy, 3267
  - clone, 3269
  - configure, 3269
  - getBackOffMultiplier, 3269
  - getCollisionAvoidancePercent, 3270
  - getInitialRedeliveryDelay, 3270
  - getMaximumRedeliveries, 3270
  - getRedeliveryDelay, 3270
  - isUseCollisionAvoidance, 3271
  - isUseExponentialBackOff, 3271
  - NO\_MAXIMUM\_REDELIVERIES, 3272
  - RedeliveryPolicy, 3269
  - setBackOffMultiplier, 3271
  - setCollisionAvoidancePercent, 3271
  - setInitialRedeliveryDelay, 3271
  - setMaximumRedeliveries, 3272
  - setUseCollisionAvoidance, 3272
  - setUseExponentialBackOff, 3272
- activemq::core::Synchronization, 3826
  - ~Synchronization, 3827
  - afterCommit, 3827
  - afterRollback, 3827
  - beforeEnd, 3827
- activemq::exceptions, 98
- activemq::exceptions::ActiveMQException, 348
  - ~ActiveMQException, 349
  - ActiveMQException, 348, 349
  - clone, 349
  - convertToCMSEException, 349
- activemq::exceptions::BrokerException, 873
  - ~BrokerException, 874
  - BrokerException, 874
  - clone, 874
- activemq::io, 98
- activemq::io::LoggingInputStream, 2474
  - ~LoggingInputStream, 2475
  - doReadArrayBounded, 2475
  - doReadByte, 2475
  - LoggingInputStream, 2475
- activemq::io::LoggingOutputStream, 2476
  - ~LoggingOutputStream, 2476
  - doWriteArrayBounded, 2477
  - doWriteByte, 2477
  - LoggingOutputStream, 2476
- activemq::library, 98
- activemq::library::ActiveMQCPP, 310
  - ~ActiveMQCPP, 311
  - ActiveMQCPP, 311

- initializeLibrary, 311
- operator=, 312
- shutdownLibrary, 312
- activemq::state, 98
- activemq::state::CommandVisitor, 1233
  - ~CommandVisitor, 1235
  - processBeginTransaction, 1235
  - processBrokerError, 1235
  - processBrokerInfo, 1236
  - processCommitTransactionOnePhase, 1236
  - processCommitTransactionTwoPhase, 1236
  - processConnectionControl, 1236
  - processConnectionError, 1236
  - processConnectionInfo, 1236
  - processConsumerControl, 1236
  - processConsumerInfo, 1237
  - processControlCommand, 1237
  - processDestinationInfo, 1237
  - processEndTransaction, 1237
  - processFlushCommand, 1237
  - processForgetTransaction, 1238
  - processKeepAliveInfo, 1238
  - processMessage, 1238
  - processMessageAck, 1238
  - processMessageDispatch, 1238
  - processMessageDispatchNotification, 1239
  - processMessagePull, 1239
  - processPrepareTransaction, 1239
  - processProducerAck, 1239
  - processProducerInfo, 1239
  - processRecoverTransactions, 1239
  - processRemoveConnection, 1239
  - processRemoveConsumer, 1239
  - processRemoveDestination, 1240
  - processRemoveInfo, 1240
  - processRemoveProducer, 1240
  - processRemoveSession, 1240
  - processRemoveSubscriptionInfo, 1240
  - processReplayCommand, 1241
  - processResponse, 1241
  - processRollbackTransaction, 1241
  - processSessionInfo, 1241
  - processShutdownInfo, 1241
  - processTransactionInfo, 1241
  - processWireFormat, 1241
- activemq::state::CommandVisitorAdapter, 1242
  - ~CommandVisitorAdapter, 1245
  - processBeginTransaction, 1245
  - processBrokerError, 1245
  - processBrokerInfo, 1245
  - processCommitTransactionOnePhase, 1245
  - processCommitTransactionTwoPhase, 1245
  - processConnectionControl, 1245
  - processConnectionError, 1245
  - processConnectionInfo, 1245
  - processConsumerControl, 1245
  - processConsumerInfo, 1245
  - processControlCommand, 1245
  - processDestinationInfo, 1245
  - processEndTransaction, 1245
  - processFlushCommand, 1245
  - processForgetTransaction, 1245
  - processKeepAliveInfo, 1245
  - processMessage, 1245
  - processMessageAck, 1245
  - processMessageDispatch, 1245
  - processMessageDispatchNotification, 1245
  - processMessagePull, 1245
  - processPrepareTransaction, 1245
  - processProducerAck, 1245
  - processProducerInfo, 1245
  - processRecoverTransactions, 1245
  - processRemoveConnection, 1245
  - processRemoveConsumer, 1245
  - processRemoveDestination, 1245
  - processRemoveInfo, 1245
  - processRemoveProducer, 1246
  - processRemoveSession, 1247
  - processRemoveSubscriptionInfo, 1247
  - processReplayCommand, 1247
  - processResponse, 1247
  - processRollbackTransaction, 1247
  - processSessionInfo, 1247
  - processShutdownInfo, 1247
  - processTransactionInfo, 1247
  - processWireFormat, 1248
- activemq::state::ConnectionState, 1429
  - ~ConnectionState, 1431
  - addSession, 1431
  - addTempDestination, 1431
  - addTransactionState, 1431
  - checkShutdown, 1431
  - ConnectionState, 1431

- getInfo, 1431
- getRecoveringPullConsumers, 1431
- getSessionState, 1431
- getSessionStates, 1431
- getTempDesinations, 1431
- getTransactionState, 1431
- getTransactionStates, 1431
- isConnectionInterruptProcessingComplete, 1432
- removeSession, 1432
- removeTempDestination, 1432
- removeTransactionState, 1432
- reset, 1432
- setConnectionInterruptProcessingComplete, 1432
- shutdown, 1432
- toString, 1432
- activemq::state::ConnectionStateTracker, 1432
  - ~ConnectionStateTracker, 1435
  - connectionInterruptProcessingComplete, 1435
  - ConnectionStateTracker, 1435
  - getMaxCacheSize, 1435
  - isRestoreConsumers, 1435
  - isRestoreProducers, 1435
  - isRestoreSessions, 1435
  - isRestoreTransaction, 1435
  - isTrackMessages, 1435
  - isTrackTransactionProducers, 1435
  - isTrackTransactions, 1435
  - processBeginTransaction, 1435
  - processCommitTransactionOnePhase, 1435
  - processCommitTransactionTwoPhase, 1436
  - processConnectionInfo, 1436
  - processConsumerInfo, 1436
  - processDestinationInfo, 1436
  - processEndTransaction, 1436
  - processMessage, 1436
  - processMessageAck, 1437
  - processPrepareTransaction, 1437
  - processProducerInfo, 1437
  - processRemoveConnection, 1437
  - processRemoveConsumer, 1437
  - processRemoveDestination, 1437
  - processRemoveProducer, 1438
  - processRemoveSession, 1438
  - processRollbackTransaction, 1438
  - processSessionInfo, 1438
  - RemoveTransactionAction, 1439
  - restore, 1438
  - setMaxCacheSize, 1439
  - setRestoreConsumers, 1439
  - setRestoreProducers, 1439
  - setRestoreSessions, 1439
  - setRestoreTransaction, 1439
  - setTrackMessages, 1439
  - setTrackTransactionProducers, 1439
  - setTrackTransactions, 1439
  - track, 1439
  - trackBack, 1439
  - transportInterrupted, 1439
- activemq::state::ConsumerState, 1535
  - ~ConsumerState, 1536
  - ConsumerState, 1536
  - getInfo, 1536
  - toString, 1536
- activemq::state::ProducerState, 3216
  - ~ProducerState, 3216
  - getInfo, 3216
  - getTransactionState, 3216
  - ProducerState, 3216
  - setTransactionState, 3216
  - toString, 3216
- activemq::state::SessionState, 3536
  - ~SessionState, 3537
  - addConsumer, 3537
  - addProducer, 3537
  - checkShutdown, 3537
  - getConsumerState, 3537
  - getConsumerStates, 3537
  - getInfo, 3537
  - getProducerState, 3537
  - getProducerStates, 3537
  - removeConsumer, 3537
  - removeProducer, 3537
  - SessionState, 3537
  - shutdown, 3537
  - toString, 3537
- activemq::state::Tracked, 3931
  - ~Tracked, 3932
  - isWaitingForResponse, 3932
  - onResponse, 3932
  - Tracked, 3932
- activemq::state::TransactionState, 3989
  - ~TransactionState, 3991
  - addCommand, 3991
  - addProducerState, 3991
  - checkShutdown, 3991

- getCommands, 3991
- getId, 3991
- getPreparedResult, 3991
- getProducerStates, 3991
- isPrepared, 3991
- setPrepared, 3991
- setPreparedResult, 3991
- shutdown, 3991
- toString, 3991
- TransactionState, 3991
- activemq::threads, 99
- activemq::threads::CompositeTask, 1255
  - ~CompositeTask, 1255
  - isPending, 1255
- activemq::threads::CompositeTaskRunner, 1256
  - ~CompositeTaskRunner, 1257
  - addTask, 1257
  - CompositeTaskRunner, 1257
  - iterate, 1257
  - removeTask, 1257
  - run, 1258
  - shutdown, 1258
  - wakeup, 1258
- activemq::threads::DedicatedTaskRunner, 1724
  - ~DedicatedTaskRunner, 1725
  - DedicatedTaskRunner, 1725
  - run, 1725
  - shutdown, 1725
  - wakeup, 1726
- activemq::threads::Task, 3846
  - ~Task, 3847
  - iterate, 3847
- activemq::threads::TaskRunner, 3849
  - ~TaskRunner, 3849
  - shutdown, 3849
  - wakeup, 3850
- activemq::transport, 99
- activemq::transport::AbstractTransportFactory, 182
  - ~AbstractTransportFactory, 183
  - createWireFormat, 183
- activemq::transport::CompositeTransport, 1259
  - ~CompositeTransport, 1259
  - addURI, 1259
  - removeURI, 1260
- activemq::transport::correlator, 100
- activemq::transport::correlator::FutureResponse, 2032
  - ~FutureResponse, 2033
  - FutureResponse, 2033
  - getResponse, 2033, 2034
  - setResponse, 2034
- activemq::transport::correlator::ResponseCorrelator, 3384
  - ~ResponseCorrelator, 3385
  - close, 3386
  - onCommand, 3386
  - oneway, 3386
  - onTransportException, 3387
  - request, 3387
  - ResponseCorrelator, 3385
  - start, 3388
- activemq::transport::DefaultTransportListener, 1757
  - ~DefaultTransportListener, 1758
  - onCommand, 1758
  - onException, 1758
  - transportInterrupted, 1758
  - transportResumed, 1758
- activemq::transport::failover, 100
- activemq::transport::failover::BackupTransport, 759
  - ~BackupTransport, 760
  - BackupTransport, 760
  - getTransport, 760
  - getUri, 760
  - isClosed, 760
  - onException, 760
  - setClosed, 761
  - setTransport, 761
  - setUri, 761
- activemq::transport::failover::BackupTransportPool, 761
  - ~BackupTransportPool, 763
  - BackupTransport, 764
  - BackupTransportPool, 763
  - getBackup, 763
  - getBackupPoolSize, 763
  - isEnabled, 763
  - isPending, 763
  - iterate, 764
  - setBackupPoolSize, 764
  - setEnabled, 764
- activemq::transport::failover::CloseTransportsTask, 1182
  - ~CloseTransportsTask, 1182
  - CloseTransportsTask, 1182
  - isPending, 1182

- iterate, 1182
- activemq::transport::failover::FailoverTransport, 1930
  - ~FailoverTransport, 1933
  - add, 1933
  - addURI, 1933
  - close, 1933
  - FailoverTransport, 1933
  - FailoverTransportListener, 1942
  - getBackOffMultiplier, 1934
  - getBackupPoolSize, 1934
  - getInitialReconnectDelay, 1934
  - getMaxCacheSize, 1934
  - getMaxReconnectAttempts, 1934
  - getMaxReconnectDelay, 1934
  - getReconnectDelay, 1934
  - getRemoteAddress, 1934
  - getStartupMaxReconnectAttempts, 1934
  - getTimeout, 1935
  - getTransportListener, 1935
  - handleTransportFailure, 1935
  - isBackup, 1935
  - isClosed, 1935
  - isConnected, 1935
  - isFaultTolerant, 1936
  - isInitialized, 1936
  - isPending, 1936
  - isRandomize, 1936
  - isTrackMessages, 1937
  - isTrackTransactionProducers, 1937
  - isUseExponentialBackOff, 1937
  - iterate, 1937
  - narrow, 1937
  - oneway, 1937
  - reconnect, 1938
  - removeURI, 1938
  - request, 1938, 1939
  - restoreTransport, 1939
  - setBackOffMultiplier, 1940
  - setBackup, 1940
  - setBackupPoolSize, 1940
  - setConnectionInterruptProcessingComplete, 1940
  - setInitialized, 1940
  - setInitialReconnectDelay, 1940
  - setMaxCacheSize, 1941
  - setMaxReconnectAttempts, 1941
  - setMaxReconnectDelay, 1941
  - setRandomize, 1941
  - setReconnectDelay, 1941
  - setStartupMaxReconnectAttempts, 1941
  - setTimeout, 1941
  - setTrackMessages, 1941
  - setTrackTransactionProducers, 1941
  - setTransportListener, 1941
  - setUseExponentialBackOff, 1941
  - setWireFormat, 1941
  - start, 1942
  - stop, 1942
- activemq::transport::failover::FailoverTransportFactory, 1942
  - ~FailoverTransportFactory, 1944
  - create, 1944
  - createComposite, 1944
  - doCreateComposite, 1944
- activemq::transport::failover::FailoverTransportListener, 1945
  - ~FailoverTransportListener, 1946
  - FailoverTransportListener, 1946
  - onCommand, 1946
  - onException, 1946
  - transportInterrupted, 1947
  - transportResumed, 1947
- activemq::transport::failover::URIPool, 4054
  - ~URIPool, 4055
  - addURI, 4055
  - addURIs, 4055
  - getURI, 4056
  - isRandomize, 4056
  - removeURI, 4056
  - setRandomize, 4056
  - URIPool, 4055
- activemq::transport::inactivity, 101
- activemq::transport::inactivity::InactivityMonitor, 2065
  - ~InactivityMonitor, 2067
  - AsyncSignalReadErrorTask, 2069
  - AsyncWriteTask, 2069
  - close, 2067
  - getInitialDelayTime, 2067
  - getReadCheckTime, 2067
  - getWriteCheckTime, 2067
  - InactivityMonitor, 2067
  - isKeepAliveResponseRequired, 2067
  - onCommand, 2067
  - oneway, 2068
  - onException, 2068
  - ReadChecker, 2069
  - setInitialDelayTime, 2068
  - setKeepAliveResponseRequired, 2069



- setReadCheckTime, 2069
  - setWriteCheckTime, 2069
  - WriteChecker, 2069
- activemq::transport::inactivity::ReadChecker, 3253
  - ~ReadChecker, 3253
  - ReadChecker, 3253
  - run, 3253
- activemq::transport::inactivity::WriteChecker, 4132
  - ~WriteChecker, 4133
  - run, 4133
  - WriteChecker, 4133
- activemq::transport::IOTransport, 2213
  - ~IOTransport, 2215
  - close, 2215
  - getRemoteAddress, 2216
  - getTransportListener, 2216
  - IOTransport, 2215
  - isClosed, 2216
  - isConnected, 2216
  - isFaultTolerant, 2216
  - narrow, 2217
  - oneway, 2217
  - reconnect, 2217
  - request, 2218
  - run, 2219
  - setInputStream, 2219
  - setOutputStream, 2219
  - setTransportListener, 2219
  - setWireFormat, 2219
  - start, 2220
  - stop, 2220
- activemq::transport::logging, 101
- activemq::transport::logging::LoggingTransport, 2477
  - ~LoggingTransport, 2478
  - LoggingTransport, 2478
  - onCommand, 2478
  - oneway, 2478
  - request, 2479
- activemq::transport::mock, 101
- activemq::transport::mock::InternalCommandListener, 2192
  - ~InternalCommandListener, 2193
  - InternalCommandListener, 2193
  - onCommand, 2193
  - run, 2193
  - setResponseBuilder, 2193
  - setTransport, 2193
- activemq::transport::mock::MockTransport, 2854
  - ~MockTransport, 2857
  - close, 2857
  - fireCommand, 2857
  - fireException, 2857
  - getInstance, 2857
  - getNumReceivedMessageBeforeFail, 2858
  - getNumReceivedMessages, 2858
  - getNumSentKeepAlives, 2858
  - getNumSentKeepAlivesBeforeFail, 2858
  - getNumSentMessageBeforeFail, 2858
  - getNumSentMessages, 2858
  - getRemoteAddress, 2858
  - getTransportListener, 2858
  - getWireFormat, 2858
  - isClosed, 2859
  - isConnected, 2859
  - isFailOnClose, 2859
  - isFailOnKeepAliveSends, 2859
  - isFailOnReceiveMessage, 2859
  - isFailOnSendMessage, 2859
  - isFailOnStart, 2859
  - isFailOnStop, 2859
  - isFaultTolerant, 2859
  - MockTransport, 2857
  - narrow, 2860
  - oneway, 2860
  - reconnect, 2860
  - request, 2861
  - setFailOnClose, 2862
  - setFailOnKeepAliveSends, 2862
  - setFailOnReceiveMessage, 2862
  - setFailOnSendMessage, 2862
  - setFailOnStart, 2862
  - setFailOnStop, 2862
  - setNumReceivedMessageBeforeFail, 2862
  - setNumReceivedMessages, 2862
  - setNumSentKeepAlives, 2862
  - setNumSentKeepAlivesBeforeFail, 2862
  - setNumSentMessageBeforeFail, 2862
  - setNumSentMessages, 2862
  - setOutgoingListener, 2862
  - setResponseBuilder, 2863
  - setTransportListener, 2863
  - setWireFormat, 2863
  - start, 2863
  - stop, 2864

- activemq::transport::mock::MockTransportFactory
  - 2864
  - ~MockTransportFactory, 2865
  - create, 2865
  - createComposite, 2865
  - doCreateComposite, 2866
- activemq::transport::mock::ResponseBuilder
  - 3382
  - ~ResponseBuilder, 3383
  - buildIncomingCommands, 3383
  - buildResponse, 3383
- activemq::transport::tcp, 102
- activemq::transport::tcp::SslTransport, 3682
  - ~SslTransport, 3683
  - configureSocket, 3683
  - createSocket, 3683
  - SslTransport, 3683
- activemq::transport::tcp::SslTransportFactory, 3684
  - ~SslTransportFactory, 3685
  - doCreateComposite, 3685
- activemq::transport::tcp::TcpTransport, 3865
  - ~TcpTransport, 3866
  - close, 3867
  - configureSocket, 3867
  - connect, 3867
  - createSocket, 3868
  - isClosed, 3868
  - isConnected, 3868
  - isFaultTolerant, 3868
  - TcpTransport, 3866
- activemq::transport::tcp::TcpTransportFactory, 3869
  - ~TcpTransportFactory, 3870
  - create, 3870
  - createComposite, 3870
  - doCreateComposite, 3870
- activemq::transport::Transport, 3996
  - ~Transport, 3998
  - getRemoteAddress, 3998
  - getTransportListener, 3998
  - isClosed, 3998
  - isConnected, 3998
  - isFaultTolerant, 3999
  - narrow, 3999
  - oneway, 3999
  - reconnect, 4000
  - request, 4000, 4001
  - setTransportListener, 4001
  - setWireFormat, 4002
- activemq::transport::mock::MockTransportFactory
  - start, 4002
  - stop, 4002
- activemq::transport::TransportFactory, 4003
  - ~TransportFactory, 4003
  - create, 4003
  - createComposite, 4004
- activemq::transport::TransportFilter, 4004
  - ~TransportFilter, 4007
  - close, 4007
  - fire, 4007
  - getRemoteAddress, 4008
  - getTransportListener, 4008
  - isClosed, 4008
  - isConnected, 4008
  - isFaultTolerant, 4009
  - listener, 4013
  - narrow, 4009
  - next, 4013
  - onCommand, 4009
  - oneway, 4010
  - onException, 4010
  - reconnect, 4010
  - request, 4011
  - setTransportListener, 4012
  - setWireFormat, 4012
  - start, 4012
  - stop, 4012
  - TransportFilter, 4007
  - transportInterrupted, 4013
  - transportResumed, 4013
- activemq::transport::TransportListener, 4013
  - ~TransportListener, 4014
  - onCommand, 4014
  - onException, 4014
  - transportInterrupted, 4015
  - transportResumed, 4015
- activemq::transport::TransportRegistry, 4015
  - ~TransportRegistry, 4016
  - findFactory, 4016
  - getInstance, 4017
  - getTransportNames, 4017
  - registerFactory, 4017
  - unregisterFactory, 4018
- activemq::util, 102
- activemq::util::ActiveMQProperties, 474
  - ~ActiveMQProperties, 476
  - ActiveMQProperties, 476
  - clear, 476
  - clone, 476
  - copy, 476

- getProperties, 476, 477
- getProperty, 477
- hasProperty, 477
- isEmpty, 478
- remove, 478
- setProperties, 478
- setProperty, 478
- toArray, 478
- toString, 479
- activemq::util::CMSExceptionSupport, 1194
  - ~CMSExceptionSupport, 1195
  - create, 1195
  - createMessageEOFException, 1195
  - createMessageFormatException, 1195
- activemq::util::CompositeData, 1253
  - ~CompositeData, 1254
  - CompositeData, 1254
  - getComponents, 1254
  - getFragment, 1254
  - getHost, 1254
  - getParameters, 1254
  - getPath, 1254
  - getScheme, 1254
  - setComponents, 1254
  - setFragment, 1254
  - setHost, 1254
  - setParameters, 1254
  - setPath, 1254
  - setScheme, 1254
  - toURI, 1254
- activemq::util::IdGenerator, 2052
  - ~IdGenerator, 2053
  - compare, 2053
  - generateId, 2053
  - getHostname, 2053
  - getSeedFromId, 2053
  - getSequenceFromId, 2053
  - IdGenerator, 2053
- activemq::util::LongSequenceGenerator, 2535
  - ~LongSequenceGenerator, 2535
  - getLastSequenceId, 2535
  - getNextSequenceId, 2535
  - LongSequenceGenerator, 2535
- activemq::util::MarshallingSupport, 2571
  - ~MarshallingSupport, 2572
  - asciiToModifiedUtf8, 2572
  - MarshallingSupport, 2572
  - modifiedUtf8ToAscii, 2573
  - readString16, 2573
  - readString32, 2573
  - writeString, 2574
  - writeString16, 2574
  - writeString32, 2575
- activemq::util::MemoryUsage, 2592
  - ~MemoryUsage, 2594
  - decreaseUsage, 2594
  - enqueueUsage, 2594
  - getLimit, 2594
  - getUsage, 2594
  - increaseUsage, 2595
  - isFull, 2595
  - MemoryUsage, 2593
  - setLimit, 2595
  - setUsage, 2595
  - waitForSpace, 2595
- activemq::util::PrimitiveList, 3067
  - ~PrimitiveList, 3070
  - getBool, 3070
  - getByte, 3071
  - getByteArray, 3071
  - getChar, 3072
  - getDouble, 3072
  - getFloat, 3072
  - getInt, 3073
  - getLong, 3073
  - getShort, 3074
  - getString, 3074
  - PrimitiveList, 3070
  - setBool, 3075
  - setByte, 3075
  - setByteArray, 3076
  - setChar, 3076
  - setDouble, 3076
  - setFloat, 3077
  - setInt, 3077
  - setLong, 3078
  - setShort, 3078
  - setString, 3078
  - toString, 3079
- activemq::util::PrimitiveMap, 3079
  - ~PrimitiveMap, 3082
  - getBool, 3082
  - getByte, 3083
  - getByteArray, 3083
  - getChar, 3084
  - getDouble, 3084
  - getFloat, 3085
  - getInt, 3085
  - getLong, 3085
  - getShort, 3086

- getString, 3086
- PrimitiveMap, 3082
- setBool, 3087
- setByte, 3087
- setByteArray, 3087
- setChar, 3088
- setDouble, 3088
- setFloat, 3088
- setInt, 3088
- setLong, 3089
- setShort, 3089
- setString, 3089
- toString, 3089
- activemq::util::PrimitiveValueConverter, 3098
  - ~PrimitiveValueConverter, 3099
  - convert, 3099
  - PrimitiveValueConverter, 3099
- activemq::util::PrimitiveValueNode, 3099
  - ~PrimitiveValueNode, 3107
  - BIG\_STRING\_TYPE, 3104
  - BOOLEAN\_TYPE, 3104
  - BYTE\_ARRAY\_TYPE, 3104
  - BYTE\_TYPE, 3104
  - CHAR\_TYPE, 3104
  - clear, 3107
  - DOUBLE\_TYPE, 3104
  - FLOAT\_TYPE, 3104
  - getBool, 3107
  - getByte, 3107
  - getByteArray, 3108
  - getChar, 3108
  - getDouble, 3108
  - getFloat, 3109
  - getInt, 3109
  - getList, 3109
  - getLong, 3109
  - getMap, 3110
  - getShort, 3110
  - getString, 3110
  - getType, 3111
  - getValue, 3111
  - INTEGER\_TYPE, 3104
  - LIST\_TYPE, 3104
  - LONG\_TYPE, 3104
  - MAP\_TYPE, 3104
  - NULL\_TYPE, 3104
  - operator=, 3111
  - operator==, 3111
  - PrimitiveType, 3104
  - PrimitiveValueNode, 3104–3107
  - setBool, 3111
  - setByte, 3112
  - setByteArray, 3112
  - setChar, 3112
  - setDouble, 3112
  - setFloat, 3112
  - setInt, 3113
  - setList, 3113
  - setLong, 3113
  - setMap, 3113
  - setShort, 3114
  - setString, 3114
  - setValue, 3114
  - SHORT\_TYPE, 3104
  - STRING\_TYPE, 3104
  - toString, 3114
- activemq::util::PrimitiveValueNode::PrimitiveValue, 3097
  - boolValue, 3098
  - byteArrayValue, 3098
  - byteValue, 3098
  - charValue, 3098
  - doubleValue, 3098
  - floatValue, 3098
  - intValue, 3098
  - listValue, 3098
  - longValue, 3098
  - mapValue, 3098
  - shortValue, 3098
  - stringValue, 3098
- activemq::util::URISupport, 4057
  - createQueryString, 4058
  - parseComposite, 4058
  - parseQuery, 4058, 4059
  - parseURL, 4059
- activemq::util::Usage, 4075
  - ~Usage, 4076
  - decreaseUsage, 4076
  - enqueueUsage, 4076
  - increaseUsage, 4077
  - isFull, 4077
  - waitForSpace, 4077
- activemq::wireformat, 103
- activemq::wireformat::MarshalAware, 2564
  - ~MarshalAware, 2565
  - afterMarshal, 2565
  - afterUnmarshal, 2565
  - beforeMarshal, 2565
  - beforeUnmarshal, 2565
  - getMarshaledForm, 2566

- isMarshalAware, 2566
- setMarshaledForm, 2566
- activemq::wireformat::openwire, 103
- activemq::wireformat::openwire::marshal, 104
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 1660, 814
  - ~BaseDataStreamMarshaller, 820
  - looseMarshal, 820
  - looseMarshalBrokerError, 820
  - looseMarshalCachedObject, 821
  - looseMarshalLong, 821
  - looseMarshalNestedObject, 822
  - looseMarshalObjectArray, 822
  - looseMarshalString, 822
  - looseUnmarshal, 823
  - looseUnmarshalBrokerError, 823
  - looseUnmarshalByteArray, 824
  - looseUnmarshalCachedObject, 824
  - looseUnmarshalConstByteArray, 824
  - looseUnmarshalLong, 825
  - looseUnmarshalNestedObject, 825
  - looseUnmarshalString, 826
  - readAsciiString, 826
  - tightMarshal1, 826
  - tightMarshal2, 827
  - tightMarshalBrokerError1, 827
  - tightMarshalBrokerError2, 828
  - tightMarshalCachedObject1, 828
  - tightMarshalCachedObject2, 828
  - tightMarshalLong1, 829
  - tightMarshalLong2, 829
  - tightMarshalNestedObject1, 830
  - tightMarshalNestedObject2, 830
  - tightMarshalObjectArray1, 831
  - tightMarshalObjectArray2, 831
  - tightMarshalString1, 831
  - tightMarshalString2, 832
  - tightUnmarshal, 832
  - tightUnmarshalBrokerError, 833
  - tightUnmarshalByteArray, 833
  - tightUnmarshalCachedObject, 833
  - tightUnmarshalConstByteArray, 834
  - tightUnmarshalLong, 834
  - tightUnmarshalNestedObject, 835
  - tightUnmarshalString, 835
  - toHexFromBytes, 836
  - toString, 836, 837
- activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1660
  - ~DataStreamMarshaller, 1661
  - createObject, 1661
  - getDataStructureType, 1668
  - looseMarshal, 1675
  - looseUnmarshal, 1683
  - tightMarshal1, 1690
  - tightMarshal2, 1698
  - tightUnmarshal, 1705
- activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3090
  - ~PrimitiveTypesMarshaller, 3092
  - marshal, 3092
  - marshalList, 3092
  - marshalMap, 3093
  - marshalPrimitive, 3093
  - marshalPrimitiveList, 3093
  - marshalPrimitiveMap, 3094
  - PrimitiveTypesMarshaller, 3092
  - unmarshal, 3094
  - unmarshalList, 3095
  - unmarshalMap, 3095
  - unmarshalPrimitive, 3096
  - unmarshalPrimitiveList, 3096
  - unmarshalPrimitiveMap, 3096
- activemq::wireformat::openwire::marshal::v1, 104
- activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller, 194
  - ~ActiveMQBlobMessageMarshaller, 195
  - ActiveMQBlobMessageMarshaller, 195
  - createObject, 195
  - getDataStructureType, 195
  - looseMarshal, 195
  - looseUnmarshal, 196
  - tightMarshal1, 196
  - tightMarshal2, 197
  - tightUnmarshal, 197
- activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller, 239
  - ~ActiveMQBytesMessageMarshaller, 240
  - ActiveMQBytesMessageMarshaller, 240
  - createObject, 240
  - getDataStructureType, 240
  - looseMarshal, 241
  - looseUnmarshal, 241
  - tightMarshal1, 241
  - tightMarshal2, 242
  - tightUnmarshal, 242

- activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, 328
  - ~ActiveMQDestinationMarshaller, 329
  - ActiveMQDestinationMarshaller, 329
  - looseMarshal, 329
  - looseUnmarshal, 329
  - tightMarshal1, 330
  - tightMarshal2, 330
  - tightUnmarshal, 331
- activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller, 369
  - ~ActiveMQMapMessageMarshaller, 370
  - ActiveMQMapMessageMarshaller, 370
  - createObject, 370
  - getDataStructureType, 370
  - looseMarshal, 370
  - looseUnmarshal, 371
  - tightMarshal1, 371
  - tightMarshal2, 372
  - tightUnmarshal, 372
- activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 397
  - ~ActiveMQMessageMarshaller, 398
  - ActiveMQMessageMarshaller, 398
  - createObject, 398
  - getDataStructureType, 398
  - looseMarshal, 399
  - looseUnmarshal, 399
  - tightMarshal1, 399
  - tightMarshal2, 400
  - tightUnmarshal, 400
- activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller, 445
  - ~ActiveMQObjectMessageMarshaller, 446
  - ActiveMQObjectMessageMarshaller, 446
  - createObject, 446
  - getDataStructureType, 446
  - looseMarshal, 446
  - looseUnmarshal, 447
  - tightMarshal1, 447
  - tightMarshal2, 448
  - tightUnmarshal, 448
- activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller, 491
  - ~ActiveMQQueueMarshaller, 492
  - ActiveMQQueueMarshaller, 492
  - createObject, 492
- activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller, 587
  - ~ActiveMQTempDestinationMarshaller, 588
  - ActiveMQTempDestinationMarshaller, 588
  - looseMarshal, 588
  - looseUnmarshal, 588
  - tightMarshal1, 589
  - tightMarshal2, 589
  - tightUnmarshal, 590
- activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 615
  - ~ActiveMQTempQueueMarshaller, 616
  - ActiveMQTempQueueMarshaller, 616
  - createObject, 616
  - getDataStructureType, 617
  - looseMarshal, 617
  - looseUnmarshal, 617
  - tightMarshal1, 618
  - tightMarshal2, 618
  - tightUnmarshal, 619
- activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller, 649
  - ~ActiveMQTempTopicMarshaller, 651
  - ActiveMQTempTopicMarshaller, 651
  - createObject, 651
  - getDataStructureType, 651
  - looseMarshal, 651
  - looseUnmarshal, 652
  - tightMarshal1, 652

- tightMarshal2, 653
- tightUnmarshal, 653
- activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller, 680
  - ~ActiveMQTextMessageMarshaller, 681
  - ActiveMQTextMessageMarshaller, 681
  - createObject, 681
  - getDataStructureType, 681
  - looseMarshal, 682
  - looseUnmarshal, 682
  - tightMarshal1, 682
  - tightMarshal2, 683
  - tightUnmarshal, 683
- activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller, 710
  - ~ActiveMQTopicMarshaller, 711
  - ActiveMQTopicMarshaller, 711
  - createObject, 711
  - getDataStructureType, 711
  - looseMarshal, 711
  - looseUnmarshal, 712
  - tightMarshal1, 712
  - tightMarshal2, 713
  - tightUnmarshal, 713
- activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller, 786
  - ~BaseCommandMarshaller, 787
  - BaseCommandMarshaller, 787
  - looseMarshal, 787
  - looseUnmarshal, 788
  - tightMarshal1, 789
  - tightMarshal2, 790
  - tightUnmarshal, 791
- activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 886
  - ~BrokerIdMarshaller, 887
  - BrokerIdMarshaller, 887
  - createObject, 887
  - getDataStructureType, 888
  - looseMarshal, 888
  - looseUnmarshal, 888
  - tightMarshal1, 889
  - tightMarshal2, 889
  - tightUnmarshal, 890
- activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 919
  - ~BrokerInfoMarshaller, 920
  - BrokerInfoMarshaller, 920
  - createObject, 920
  - getDataStructureType, 920
  - looseMarshal, 921
  - looseUnmarshal, 921
  - tightMarshal1, 921
  - tightMarshal2, 922
  - tightUnmarshal, 922
- activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller, 1315
  - ~ConnectionControlMarshaller, 1316
  - ConnectionControlMarshaller, 1316
  - createObject, 1316
  - getDataStructureType, 1317
  - looseMarshal, 1317
  - looseUnmarshal, 1317
  - tightMarshal1, 1318
  - tightMarshal2, 1318
  - tightUnmarshal, 1319
- activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1349
  - ~ConnectionErrorMarshaller, 1350
  - ConnectionErrorMarshaller, 1350
  - createObject, 1350
  - getDataStructureType, 1350
  - looseMarshal, 1350
  - looseUnmarshal, 1351
  - tightMarshal1, 1351
  - tightMarshal2, 1352
  - tightUnmarshal, 1352
- activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller, 1381
  - ~ConnectionIdMarshaller, 1382
  - ConnectionIdMarshaller, 1382
  - createObject, 1382
  - getDataStructureType, 1382
  - looseMarshal, 1382
  - looseUnmarshal, 1383
  - tightMarshal1, 1383
  - tightMarshal2, 1384
  - tightUnmarshal, 1384
- activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller, 1412
  - ~ConnectionInfoMarshaller, 1414
  - ConnectionInfoMarshaller, 1414
  - createObject, 1414
  - getDataStructureType, 1414
  - looseMarshal, 1414
  - looseUnmarshal, 1415
  - tightMarshal1, 1415
  - tightMarshal2, 1416
  - tightUnmarshal, 1416

- activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller, 1459
  - ~ConsumerControlMarshaller, 1461
  - ConsumerControlMarshaller, 1461
  - createObject, 1461
  - getDataStructureType, 1461
  - looseMarshal, 1461
  - looseUnmarshal, 1462
  - tightMarshal1, 1462
  - tightMarshal2, 1463
  - tightUnmarshal, 1463
- activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller, 1489
  - ~ConsumerIdMarshaller, 1490
  - ConsumerIdMarshaller, 1490
  - createObject, 1490
  - getDataStructureType, 1491
  - looseMarshal, 1491
  - looseUnmarshal, 1491
  - tightMarshal1, 1492
  - tightMarshal2, 1492
  - tightUnmarshal, 1493
- activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller, 1522
  - ~ConsumerInfoMarshaller, 1524
  - ConsumerInfoMarshaller, 1524
  - createObject, 1524
  - getDataStructureType, 1524
  - looseMarshal, 1524
  - looseUnmarshal, 1525
  - tightMarshal1, 1525
  - tightMarshal2, 1526
  - tightUnmarshal, 1526
- activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller, 1552
  - ~ControlCommandMarshaller, 1553
  - ControlCommandMarshaller, 1553
  - createObject, 1553
  - getDataStructureType, 1554
  - looseMarshal, 1554
  - looseUnmarshal, 1554
  - tightMarshal1, 1555
  - tightMarshal2, 1555
  - tightUnmarshal, 1556
- activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller, 1587
  - ~DataArrayResponseMarshaller, 1589
  - createObject, 1589
  - DataArrayResponseMarshaller, 1589
  - getDataStructureType, 1589
- activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1589
  - looseUnmarshal, 1590
  - tightMarshal1, 1590
  - tightMarshal2, 1591
  - tightUnmarshal, 1591
- activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1656
  - ~DataResponseMarshaller, 1657
  - createObject, 1657
  - DataResponseMarshaller, 1657
  - getDataStructureType, 1657
  - looseMarshal, 1658
  - looseUnmarshal, 1658
  - tightMarshal1, 1659
  - tightMarshal2, 1659
  - tightUnmarshal, 1660
- activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller, 1797
  - ~DestinationInfoMarshaller, 1798
  - createObject, 1798
  - DestinationInfoMarshaller, 1798
  - getDataStructureType, 1799
  - looseMarshal, 1799
  - looseUnmarshal, 1799
  - tightMarshal1, 1800
  - tightMarshal2, 1800
  - tightUnmarshal, 1801
- activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1831
  - ~DiscoveryEventMarshaller, 1832
  - createObject, 1832
  - DiscoveryEventMarshaller, 1832
  - getDataStructureType, 1833
  - looseMarshal, 1833
  - looseUnmarshal, 1833
  - tightMarshal1, 1834
  - tightMarshal2, 1834
  - tightUnmarshal, 1835
- activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller, 1919
  - ~ExceptionResponseMarshaller, 1920
  - createObject, 1920
  - ExceptionResponseMarshaller, 1920
  - getDataStructureType, 1920
  - looseMarshal, 1921
  - looseUnmarshal, 1921
  - tightMarshal1, 1922
  - tightMarshal2, 1922
  - tightUnmarshal, 1923



- activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller, 2301
  - 2019
  - ~FlushCommandMarshaller, 2020
  - createObject, 2020
  - FlushCommandMarshaller, 2020
  - getDataStructureType, 2020
  - looseMarshal, 2020
  - looseUnmarshal, 2021
  - tightMarshal1, 2021
  - tightMarshal2, 2022
  - tightUnmarshal, 2022
- activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller, 2180
  - ~IntegerResponseMarshaller, 2181
  - createObject, 2181
  - getDataStructureType, 2181
  - IntegerResponseMarshaller, 2181
  - looseMarshal, 2181
  - looseUnmarshal, 2182
  - tightMarshal1, 2182
  - tightMarshal2, 2183
  - tightUnmarshal, 2183
- activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller, 2248
  - ~JournalQueueAckMarshaller, 2249
  - createObject, 2249
  - getDataStructureType, 2249
  - JournalQueueAckMarshaller, 2249
  - looseMarshal, 2249
  - looseUnmarshal, 2250
  - tightMarshal1, 2250
  - tightMarshal2, 2251
  - tightUnmarshal, 2251
- activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller, 2277
  - ~JournalTopicAckMarshaller, 2278
  - createObject, 2278
  - getDataStructureType, 2278
  - JournalTopicAckMarshaller, 2278
  - looseMarshal, 2278
  - looseUnmarshal, 2279
  - tightMarshal1, 2279
  - tightMarshal2, 2280
  - tightUnmarshal, 2280
- activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller, 2300
  - ~JournalTraceMarshaller, 2301
  - createObject, 2301
  - getDataStructureType, 2301
  - JournalTraceMarshaller, 2301
- activemq::wireformat::openwire::marshal::v1::JobCommandMarshaller, 2301
  - looseUnmarshal, 2302
  - tightMarshal1, 2302
  - tightMarshal2, 2303
  - tightUnmarshal, 2303
- activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller, 2331
  - ~JournalTransactionMarshaller, 2332
  - createObject, 2332
  - getDataStructureType, 2333
  - JournalTransactionMarshaller, 2332
- activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller, 2359
  - ~KeepAliveInfoMarshaller, 2361
  - createObject, 2361
  - getDataStructureType, 2361
  - KeepAliveInfoMarshaller, 2361
  - looseMarshal, 2361
  - looseUnmarshal, 2362
  - tightMarshal1, 2362
  - tightMarshal2, 2363
  - tightUnmarshal, 2363
- activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller, 2395
  - ~LastPartialCommandMarshaller, 2396
  - createObject, 2396
  - getDataStructureType, 2396
  - LastPartialCommandMarshaller, 2396
  - looseMarshal, 2397
  - looseUnmarshal, 2397
  - tightMarshal1, 2398
  - tightMarshal2, 2398
  - tightUnmarshal, 2399
- activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller, 2446
  - ~LocalTransactionIdMarshaller, 2447
  - createObject, 2447
  - getDataStructureType, 2447
  - LocalTransactionIdMarshaller, 2447
  - looseMarshal, 2448
  - looseUnmarshal, 2448
  - tightMarshal1, 2448
  - tightMarshal2, 2449
  - tightUnmarshal, 2449

- activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller, 2798
  - ~MessageAckMarshaller, 2798
  - looseMarshal, 2799
- activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller, 2800
  - MessageMarshaller, 2799
  - tightMarshal1, 2800
  - tightMarshal2, 2801
  - tightUnmarshal, 2802
- activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller, 2845
  - ~MessagePullMarshaller, 2847
  - createObject, 2847
  - getDataStructureType, 2847
  - looseMarshal, 2847
- activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2848
  - MessagePullMarshaller, 2847
  - tightMarshal1, 2848
  - tightMarshal2, 2849
  - tightUnmarshal, 2849
- activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilter, 2901
  - ~NetworkBridgeFilterMarshaller, 2902
  - createObject, 2902
  - getDataStructureType, 2902
  - looseMarshal, 2902
- activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller, 2903
  - NetworkBridgeFilterMarshaller, 2902
  - tightMarshal1, 2903
  - tightMarshal2, 2904
  - tightUnmarshal, 2904
- activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller, 3028
  - ~PartialCommandMarshaller, 3029
  - createObject, 3029
  - getDataStructureType, 3029
  - looseMarshal, 3030
  - looseUnmarshal, 3030
  - PartialCommandMarshaller, 3029
- activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 3031
  - tightMarshal2, 3031
  - tightUnmarshal, 3032
- activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller, 3150
  - ~ProducerAckMarshaller, 3151
  - createObject, 3151
  - getDataStructureType, 3151
  - looseMarshal, 3151
  - looseUnmarshal, 3152
  - ProducerAckMarshaller, 3151
- activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 2570
  - ~MarshallerFactory, 2570
  - configure, 2570
- activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller, 2665
  - ~MessageAckMarshaller, 2667
  - createObject, 2667
  - getDataStructureType, 2667
  - looseMarshal, 2667
  - looseUnmarshal, 2668
  - MessageAckMarshaller, 2667
  - tightMarshal1, 2668
  - tightMarshal2, 2669
  - tightUnmarshal, 2669
- activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2707
  - ~MessageDispatchMarshaller, 2709
  - createObject, 2709
  - getDataStructureType, 2709
  - looseMarshal, 2709
  - looseUnmarshal, 2710
  - MessageDispatchMarshaller, 2709
  - tightMarshal1, 2710
  - tightMarshal2, 2711
  - tightUnmarshal, 2711
- activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller, 2737
  - ~MessageDispatchNotificationMarshaller, 2739
  - createObject, 2739
  - getDataStructureType, 2739
  - looseMarshal, 2739
  - looseUnmarshal, 2740
  - MessageDispatchNotificationMarshaller, 2739
  - tightMarshal1, 2740
  - tightMarshal2, 2741
  - tightUnmarshal, 2741
- activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 2775
  - ~MessageIdMarshaller, 2776
  - createObject, 2776
  - getDataStructureType, 2777
  - looseMarshal, 2777
  - looseUnmarshal, 2777
  - MessageIdMarshaller, 2776
  - tightMarshal1, 2778
  - tightMarshal2, 2778
  - tightUnmarshal, 2779

- tightMarshal1, 3152
- tightMarshal2, 3153
- tightUnmarshal, 3153
- activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller, 3181
  - ~ProducerIdMarshaller, 3182
  - createObject, 3182
  - getDataStructureType, 3183
  - looseMarshal, 3183
  - looseUnmarshal, 3183
  - ProducerIdMarshaller, 3182
  - tightMarshal1, 3184
  - tightMarshal2, 3184
  - tightUnmarshal, 3185
- activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller, 3199
  - ~ProducerInfoMarshaller, 3200
  - createObject, 3200
  - getDataStructureType, 3200
  - looseMarshal, 3200
  - looseUnmarshal, 3201
  - ProducerInfoMarshaller, 3200
  - tightMarshal1, 3201
  - tightMarshal2, 3202
  - tightUnmarshal, 3202
- activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller, 3300
  - ~RemoveInfoMarshaller, 3302
  - createObject, 3302
  - getDataStructureType, 3302
  - looseMarshal, 3302
  - looseUnmarshal, 3303
  - RemoveInfoMarshaller, 3302
  - tightMarshal1, 3303
  - tightMarshal2, 3304
  - tightUnmarshal, 3304
- activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller, 3318
  - ~RemoveSubscriptionInfoMarshaller, 3319
  - createObject, 3319
  - getDataStructureType, 3319
  - looseMarshal, 3320
  - looseUnmarshal, 3320
  - RemoveSubscriptionInfoMarshaller, 3319
  - tightMarshal1, 3321
  - tightMarshal2, 3321
  - tightUnmarshal, 3321
- activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller, 3351
  - ~ReplayCommandMarshaller, 3352
  - createObject, 3352
  - getDataStructureType, 3353
  - looseMarshal, 3353
  - looseUnmarshal, 3353
  - ReplayCommandMarshaller, 3352
  - tightMarshal1, 3354
  - tightMarshal2, 3354
  - tightUnmarshal, 3355
- activemq::wireformat::openwire::marshal::v1::ResponseMarshaller, 3408
  - ~ResponseMarshaller, 3410
  - createObject, 3410
  - getDataStructureType, 3410
  - looseMarshal, 3410
  - looseUnmarshal, 3411
  - ResponseMarshaller, 3410
  - tightMarshal1, 3411
  - tightMarshal2, 3412
  - tightUnmarshal, 3413
- activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller, 3501
  - ~SessionIdMarshaller, 3502
  - createObject, 3502
  - getDataStructureType, 3502
  - looseMarshal, 3502
  - looseUnmarshal, 3503
  - SessionIdMarshaller, 3502
  - tightMarshal1, 3503
  - tightMarshal2, 3504
  - tightUnmarshal, 3504
- activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller, 3517
  - ~SessionInfoMarshaller, 3518
  - createObject, 3518
  - getDataStructureType, 3518
  - looseMarshal, 3518
  - looseUnmarshal, 3519
  - SessionInfoMarshaller, 3518
  - tightMarshal1, 3519
  - tightMarshal2, 3520
  - tightUnmarshal, 3520
- activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller, 3584
  - ~ShutdownInfoMarshaller, 3586
  - createObject, 3586
  - getDataStructureType, 3586
  - looseMarshal, 3586
  - looseUnmarshal, 3587
  - ShutdownInfoMarshaller, 3586

- tightMarshal1, 3587
- tightMarshal2, 3588
- tightUnmarshal, 3588
- activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller, 3791
  - ~SubscriptionInfoMarshaller, 3792
  - createObject, 3792
  - getDataStructureType, 3792
  - looseMarshal, 3792
  - looseUnmarshal, 3793
  - SubscriptionInfoMarshaller, 3792
  - tightMarshal1, 3793
  - tightMarshal2, 3794
  - tightUnmarshal, 3794
- activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller, 3939
  - ~TransactionIdMarshaller, 3941
  - looseMarshal, 3941
  - looseUnmarshal, 3941
  - tightMarshal1, 3942
  - tightMarshal2, 3942
  - tightUnmarshal, 3943
  - TransactionIdMarshaller, 3941
- activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller, 3968
  - ~TransactionInfoMarshaller, 3969
  - createObject, 3969
  - getDataStructureType, 3970
  - looseMarshal, 3970
  - looseUnmarshal, 3970
  - tightMarshal1, 3971
  - tightMarshal2, 3971
  - tightUnmarshal, 3972
  - TransactionInfoMarshaller, 3969
- activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller, 4121
  - ~WireFormatInfoMarshaller, 4122
  - createObject, 4122
  - getDataStructureType, 4122
  - looseMarshal, 4122
  - looseUnmarshal, 4123
  - tightMarshal1, 4123
  - tightMarshal2, 4124
  - tightUnmarshal, 4124
  - WireFormatInfoMarshaller, 4122
- activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller, 4160
  - ~XATransactionIdMarshaller, 4161
  - createObject, 4161
  - getDataStructureType, 4161
- looseMarshal, 4161
- looseUnmarshal, 4162
- tightMarshal1, 4162
- tightUnmarshal, 4163
- XATransactionIdMarshaller, 4161
- activemq::wireformat::openwire::marshal::v2, 109
- activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller, 202
  - ~ActiveMQBlobMessageMarshaller, 203
  - ActiveMQBlobMessageMarshaller, 203
  - createObject, 203
  - getDataStructureType, 204
  - looseMarshal, 204
  - looseUnmarshal, 204
  - tightMarshal1, 205
  - tightMarshal2, 205
  - tightUnmarshal, 206
- activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller, 256
  - ~ActiveMQBytesMessageMarshaller, 257
  - ActiveMQBytesMessageMarshaller, 257
  - createObject, 257
  - getDataStructureType, 257
  - looseMarshal, 258
  - looseUnmarshal, 258
  - tightMarshal1, 258
  - tightMarshal2, 259
  - tightUnmarshal, 259
- activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller, 340
  - ~ActiveMQDestinationMarshaller, 341
  - ActiveMQDestinationMarshaller, 341
  - looseMarshal, 341
  - looseUnmarshal, 341
  - tightMarshal1, 342
  - tightMarshal2, 342
  - tightUnmarshal, 343
- activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller, 381
  - ~ActiveMQMapMessageMarshaller, 383
  - ActiveMQMapMessageMarshaller, 383
  - createObject, 383
  - getDataStructureType, 383
  - looseMarshal, 383
  - looseUnmarshal, 384

- tightMarshal1, 384
- tightMarshal2, 385
- tightUnmarshal, 385
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller, 410
  - ~ActiveMQMessageMarshaller, 411
  - ActiveMQMessageMarshaller, 411
  - createObject, 411
  - getDataSetType, 411
  - looseMarshal, 411
  - looseUnmarshal, 412
  - tightMarshal1, 412
  - tightMarshal2, 413
  - tightUnmarshal, 413
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller, 457
  - ~ActiveMQObjectMessageMarshaller, 459
  - ActiveMQObjectMessageMarshaller, 459
  - createObject, 459
  - getDataSetType, 459
  - looseMarshal, 459
  - looseUnmarshal, 460
  - tightMarshal1, 460
  - tightMarshal2, 461
  - tightUnmarshal, 461
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller, 504
  - ~ActiveMQQueueMarshaller, 505
  - ActiveMQQueueMarshaller, 505
  - createObject, 505
  - getDataSetType, 505
  - looseMarshal, 505
  - looseUnmarshal, 506
  - tightMarshal1, 506
  - tightMarshal2, 507
  - tightUnmarshal, 507
- activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller, 570
  - ~ActiveMQStreamMessageMarshaller, 571
  - ActiveMQStreamMessageMarshaller, 571
  - createObject, 571
  - getDataSetType, 571
  - looseMarshal, 572
  - looseUnmarshal, 572
  - tightMarshal1, 572
  - tightMarshal2, 573
- tightUnmarshal, 573
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller, 599
  - ActiveMQTempDestinationMarshaller, 600
  - looseMarshal, 600
  - looseUnmarshal, 600
  - tightMarshal1, 601
  - tightMarshal2, 601
  - tightUnmarshal, 602
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller, 628
  - ActiveMQTempQueueMarshaller, 629
  - createObject, 629
  - getDataSetType, 629
  - looseMarshal, 630
  - looseUnmarshal, 630
  - tightMarshal1, 630
  - tightMarshal2, 631
  - tightUnmarshal, 631
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller, 658
  - ~ActiveMQTempTopicMarshaller, 659
  - ActiveMQTempTopicMarshaller, 659
  - createObject, 659
  - getDataSetType, 659
  - looseMarshal, 660
  - looseUnmarshal, 660
  - tightMarshal1, 660
  - tightMarshal2, 661
  - tightUnmarshal, 661
- activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller, 693
  - ~ActiveMQTextMessageMarshaller, 694
  - createObject, 694
  - getDataSetType, 694
  - looseMarshal, 694
  - looseUnmarshal, 695
  - tightMarshal1, 695
  - tightMarshal2, 696
  - tightUnmarshal, 696
- activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller, 722
  - ~ActiveMQTopicMarshaller, 724
  - ActiveMQTopicMarshaller, 724

- createObject, 724
- getDataStructureType, 724
- looseMarshal, 724
- looseUnmarshal, 725
- tightMarshal1, 725
- tightMarshal2, 726
- tightUnmarshal, 726
- activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller, 807
  - ~BaseCommandMarshaller, 808
  - BaseCommandMarshaller, 808
  - looseMarshal, 808
  - looseUnmarshal, 809
  - tightMarshal1, 810
  - tightMarshal2, 811
  - tightUnmarshal, 812
- activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller, 898
  - ~BrokerIdMarshaller, 899
  - BrokerIdMarshaller, 899
  - createObject, 899
  - getDataStructureType, 900
  - looseMarshal, 900
  - looseUnmarshal, 900
  - tightMarshal1, 901
  - tightMarshal2, 901
  - tightUnmarshal, 902
- activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller, 932
  - ~BrokerInfoMarshaller, 933
  - BrokerInfoMarshaller, 933
  - createObject, 933
  - getDataStructureType, 933
  - looseMarshal, 933
  - looseUnmarshal, 934
  - tightMarshal1, 934
  - tightMarshal2, 935
  - tightUnmarshal, 935
- activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller, 1328
  - ~ConnectionControlMarshaller, 1329
  - ConnectionControlMarshaller, 1329
  - createObject, 1329
  - getDataStructureType, 1329
  - looseMarshal, 1330
  - looseUnmarshal, 1330
  - tightMarshal1, 1330
  - tightMarshal2, 1331
  - tightUnmarshal, 1331
- activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller, 1336
  - ~ConnectionErrorMarshaller, 1337
  - ConnectionErrorMarshaller, 1337
  - createObject, 1337
  - getDataStructureType, 1337
  - looseMarshal, 1338
  - looseUnmarshal, 1338
  - tightMarshal1, 1338
  - tightMarshal2, 1339
  - tightUnmarshal, 1339
- activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller, 1368
  - ~ConnectionIdMarshaller, 1370
  - ConnectionIdMarshaller, 1370
  - createObject, 1370
  - getDataStructureType, 1370
  - looseMarshal, 1370
  - looseUnmarshal, 1371
  - tightMarshal1, 1371
  - tightMarshal2, 1371
  - tightUnmarshal, 1372
- activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller, 1400
  - ~ConnectionInfoMarshaller, 1401
  - ConnectionInfoMarshaller, 1401
  - createObject, 1401
  - getDataStructureType, 1401
  - looseMarshal, 1401
  - looseUnmarshal, 1402
  - tightMarshal1, 1402
  - tightMarshal2, 1403
  - tightUnmarshal, 1403
- activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller, 1447
  - ~ConsumerControlMarshaller, 1448
  - ConsumerControlMarshaller, 1448
  - createObject, 1448
  - getDataStructureType, 1448
  - looseMarshal, 1448
  - looseUnmarshal, 1449
  - tightMarshal1, 1449
  - tightMarshal2, 1450
  - tightUnmarshal, 1450
- activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller, 1477
  - ~ConsumerIdMarshaller, 1478
  - ConsumerIdMarshaller, 1478
  - createObject, 1478
  - getDataStructureType, 1478

- looseMarshal, 1479
- looseUnmarshal, 1479
- tightMarshal1, 1479
- tightMarshal2, 1480
- tightUnmarshal, 1480
- activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller, 1510
  - ~ConsumerInfoMarshaller, 1511
  - ConsumerInfoMarshaller, 1511
  - createObject, 1511
  - getDataStructureType, 1511
  - looseMarshal, 1511
  - looseUnmarshal, 1512
  - tightMarshal1, 1512
  - tightMarshal2, 1513
  - tightUnmarshal, 1513
- activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller, 1539
  - ~ControlCommandMarshaller, 1541
  - ControlCommandMarshaller, 1541
  - createObject, 1541
  - getDataStructureType, 1541
  - looseMarshal, 1541
  - looseUnmarshal, 1542
  - tightMarshal1, 1542
  - tightMarshal2, 1543
  - tightUnmarshal, 1543
- activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller, 1574
  - ~DataArrayResponseMarshaller, 1576
  - createObject, 1576
  - DataArrayResponseMarshaller, 1576
  - getDataStructureType, 1576
  - looseMarshal, 1576
  - looseUnmarshal, 1577
  - tightMarshal1, 1577
  - tightMarshal2, 1578
  - tightUnmarshal, 1578
- activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller, 1643
  - ~DataResponseMarshaller, 1644
  - createObject, 1644
  - DataResponseMarshaller, 1644
  - getDataStructureType, 1645
  - looseMarshal, 1645
  - looseUnmarshal, 1645
  - tightMarshal1, 1646
  - tightMarshal2, 1646
  - tightUnmarshal, 1647
- activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller, 1784
  - ~DestinationInfoMarshaller, 1786
  - createObject, 1786
  - DestinationInfoMarshaller, 1786
  - getDataStructureType, 1786
  - looseMarshal, 1786
  - looseUnmarshal, 1787
  - tightMarshal1, 1787
  - tightMarshal2, 1788
  - tightUnmarshal, 1788
- activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, 1819
  - ~DiscoveryEventMarshaller, 1820
  - createObject, 1820
  - DiscoveryEventMarshaller, 1820
  - getDataStructureType, 1821
  - looseMarshal, 1821
  - looseUnmarshal, 1821
  - tightMarshal1, 1822
  - tightMarshal2, 1822
  - tightUnmarshal, 1823
- activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller, 1902
  - ~ExceptionResponseMarshaller, 1903
  - createObject, 1903
  - ExceptionResponseMarshaller, 1903
  - getDataStructureType, 1903
  - DataArrayResponseMarshaller, 1904
  - looseMarshal, 1904
  - looseUnmarshal, 1904
  - tightMarshal1, 1904
  - tightMarshal2, 1905
  - tightUnmarshal, 1905
- activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller, 2006
  - ~FlushCommandMarshaller, 2007
  - createObject, 2007
  - FlushCommandMarshaller, 2007
  - getDataStructureType, 2007
  - looseMarshal, 2008
  - looseUnmarshal, 2008
  - tightMarshal1, 2008
  - tightMarshal2, 2009
  - tightUnmarshal, 2009
- activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller, 2167
  - ~IntegerResponseMarshaller, 2168
  - createObject, 2168
  - getDataStructureType, 2168
  - IntegerResponseMarshaller, 2168

- looseMarshal, 2168
- looseUnmarshal, 2169
- tightMarshal1, 2169
- tightMarshal2, 2170
- tightUnmarshal, 2170
- activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller, 2232
  - ~JournalQueueAckMarshaller, 2233
  - createObject, 2233
  - getDataStructureType, 2233
  - JournalQueueAckMarshaller, 2233
  - looseMarshal, 2233
  - looseUnmarshal, 2234
  - tightMarshal1, 2234
  - tightMarshal2, 2235
  - tightUnmarshal, 2235
- activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller, 2261
  - ~JournalTopicAckMarshaller, 2262
  - createObject, 2262
  - getDataStructureType, 2262
  - JournalTopicAckMarshaller, 2262
  - looseMarshal, 2262
  - looseUnmarshal, 2263
  - tightMarshal1, 2263
  - tightMarshal2, 2264
  - tightUnmarshal, 2264
- activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller, 2283
  - ~JournalTraceMarshaller, 2285
  - createObject, 2285
  - getDataStructureType, 2285
  - JournalTraceMarshaller, 2285
  - looseMarshal, 2285
  - looseUnmarshal, 2286
  - tightMarshal1, 2286
  - tightMarshal2, 2286
  - tightUnmarshal, 2287
- activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller, 2315
  - ~JournalTransactionMarshaller, 2316
  - createObject, 2316
  - getDataStructureType, 2317
  - JournalTransactionMarshaller, 2316
  - looseMarshal, 2317
  - looseUnmarshal, 2317
  - tightMarshal1, 2318
  - tightMarshal2, 2318
  - tightUnmarshal, 2319
- activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller, 2342
  - ~KeepAliveInfoMarshaller, 2344
  - createObject, 2344
  - getDataStructureType, 2344
  - KeepAliveInfoMarshaller, 2344
  - looseMarshal, 2344
  - looseUnmarshal, 2345
  - tightMarshal1, 2345
  - tightMarshal2, 2346
  - tightUnmarshal, 2346
- activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller, 2382
  - ~LastPartialCommandMarshaller, 2383
  - createObject, 2383
  - getDataStructureType, 2384
  - LastPartialCommandMarshaller, 2383
  - looseMarshal, 2384
  - looseUnmarshal, 2384
  - tightMarshal1, 2385
  - tightMarshal2, 2385
  - tightUnmarshal, 2386
- activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller, 2429
  - ~LocalTransactionIdMarshaller, 2430
  - createObject, 2430
  - getDataStructureType, 2430
  - LocalTransactionIdMarshaller, 2430
  - looseMarshal, 2431
  - looseUnmarshal, 2431
  - tightMarshal1, 2431
  - tightMarshal2, 2432
  - tightUnmarshal, 2432
- activemq::wireformat::openwire::marshal::v2::MarshallerFactory, 2570
  - ~MarshallerFactory, 2571
  - configure, 2571
- activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller, 2653
  - ~MessageAckMarshaller, 2654
  - createObject, 2654
  - getDataStructureType, 2654
  - looseMarshal, 2654
  - looseUnmarshal, 2655
  - MessageAckMarshaller, 2654
  - tightMarshal1, 2655
  - tightMarshal2, 2656
  - tightUnmarshal, 2656
- activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller, 2690



- ~MessageDispatchMarshaller, 2692
- createObject, 2692
- getDataSetType, 2692
- looseMarshal, 2692
- looseUnmarshal, 2693
- MessageDispatchMarshaller, 2692
- tightMarshal1, 2693
- tightMarshal2, 2694
- tightUnmarshal, 2694
- activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller, 2725
  - ~MessageDispatchNotificationMarshaller, 2726
  - createObject, 2726
  - getDataSetType, 2726
  - looseMarshal, 2726
  - looseUnmarshal, 2727
  - MessageDispatchNotificationMarshaller, 2726
  - tightMarshal1, 2727
  - tightMarshal2, 2728
  - tightUnmarshal, 2728
- activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller, 2755
  - ~MessageIdMarshaller, 2756
  - createObject, 2756
  - getDataSetType, 2756
  - looseMarshal, 2757
  - looseUnmarshal, 2757
  - MessageIdMarshaller, 2756
  - tightMarshal1, 2758
  - tightMarshal2, 2758
  - tightUnmarshal, 2758
- activemq::wireformat::openwire::marshal::v2::MessageMarshaller, 2789
  - ~MessageMarshaller, 2790
  - looseMarshal, 2790
  - looseUnmarshal, 2791
  - MessageMarshaller, 2790
  - tightMarshal1, 2791
  - tightMarshal2, 2792
  - tightUnmarshal, 2793
- activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller, 2828
  - ~MessagePullMarshaller, 2830
  - createObject, 2830
  - getDataSetType, 2830
  - looseMarshal, 2830
  - looseUnmarshal, 2831
  - MessagePullMarshaller, 2830
  - tightMarshal1, 2831
  - tightMarshal2, 2832
  - tightUnmarshal, 2832
- activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller, 2881
  - ~NetworkBridgeFilterMarshaller, 2882
  - createObject, 2882
  - getDataSetType, 2882
  - looseMarshal, 2882
  - MessageDispatchNotificationMarshaller, 2883
  - NetworkBridgeFilterMarshaller, 2882
  - tightMarshal1, 2883
  - tightMarshal2, 2884
  - tightUnmarshal, 2884
- activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller, 3010
  - ~PartialCommandMarshaller, 3011
  - createObject, 3011
  - getDataSetType, 3011
  - looseMarshal, 3012
  - looseUnmarshal, 3012
  - PartialCommandMarshaller, 3011
  - tightMarshal1, 3013
  - tightMarshal2, 3013
  - tightUnmarshal, 3014
- activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller, 3128
  - ~ProducerAckMarshaller, 3130
  - createObject, 3130
  - getDataSetType, 3130
  - looseMarshal, 3130
  - looseUnmarshal, 3131
  - ProducerAckMarshaller, 3130
  - tightMarshal1, 3131
  - tightMarshal2, 3132
  - tightUnmarshal, 3132
- activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller, 3161
  - ~ProducerIdMarshaller, 3162
  - createObject, 3162
  - getDataSetType, 3162
  - looseMarshal, 3163
  - ProducerIdMarshaller, 3163
  - tightMarshal1, 3164
  - tightMarshal2, 3164
  - tightUnmarshal, 3164
- activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller, 3194
  - ~ProducerInfoMarshaller, 3196

- createObject, 3196
- getDataStructureType, 3196
- looseMarshal, 3196
- looseUnmarshal, 3197
- ProducerInfoMarshaller, 3196
- tightMarshal1, 3197
- tightMarshal2, 3198
- tightUnmarshal, 3198
- activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller, 3288
  - ~RemoveInfoMarshaller, 3289
  - createObject, 3289
  - getDataStructureType, 3289
  - looseMarshal, 3289
  - looseUnmarshal, 3290
  - RemoveInfoMarshaller, 3289
  - tightMarshal1, 3290
  - tightMarshal2, 3291
  - tightUnmarshal, 3291
- activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller, 3326
  - ~RemoveSubscriptionInfoMarshaller, 3328
  - createObject, 3328
  - getDataStructureType, 3328
  - looseMarshal, 3328
  - looseUnmarshal, 3329
  - RemoveSubscriptionInfoMarshaller, 3328
  - tightMarshal1, 3329
  - tightMarshal2, 3330
  - tightUnmarshal, 3330
- activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller, 3355
  - ~ReplayCommandMarshaller, 3357
  - createObject, 3357
  - getDataStructureType, 3357
  - looseMarshal, 3357
  - looseUnmarshal, 3358
  - ReplayCommandMarshaller, 3357
  - tightMarshal1, 3358
  - tightMarshal2, 3359
  - tightUnmarshal, 3359
- activemq::wireformat::openwire::marshal::v2::ResponseMarshaller, 3393
  - ~ResponseMarshaller, 3395
  - createObject, 3395
  - getDataStructureType, 3395
  - looseMarshal, 3395
  - looseUnmarshal, 3396
  - ResponseMarshaller, 3395
  - tightMarshal1, 3396
  - tightMarshal2, 3397
  - tightUnmarshal, 3398
- activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller, 3481
  - ~SessionIdMarshaller, 3482
  - createObject, 3482
  - getDataStructureType, 3482
  - looseMarshal, 3482
  - looseUnmarshal, 3483
  - SessionIdMarshaller, 3482
  - tightMarshal1, 3483
  - tightMarshal2, 3484
  - tightUnmarshal, 3484
- activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller, 3525
  - ~SessionInfoMarshaller, 3527
  - createObject, 3527
  - getDataStructureType, 3527
  - looseMarshal, 3527
  - looseUnmarshal, 3528
  - SessionInfoMarshaller, 3527
  - tightMarshal1, 3528
  - tightMarshal2, 3529
  - tightUnmarshal, 3529
- activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller, 3580
  - ~ShutdownInfoMarshaller, 3581
  - createObject, 3581
  - getDataStructureType, 3582
  - looseMarshal, 3582
  - looseUnmarshal, 3583
  - ShutdownInfoMarshaller, 3581
  - tightMarshal1, 3583
  - tightMarshal2, 3583
  - tightUnmarshal, 3584
- activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller, 3807
  - ~SubscriptionInfoMarshaller, 3808
  - createObject, 3808
  - getDataStructureType, 3808
  - looseMarshal, 3808
  - looseUnmarshal, 3809
  - SubscriptionInfoMarshaller, 3808
  - tightMarshal1, 3809
  - tightMarshal2, 3810
  - tightUnmarshal, 3810
- activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller, 3943
  - ~TransactionIdMarshaller, 3945

- looseMarshal, 3945
- looseUnmarshal, 3945
- tightMarshal1, 3946
- tightMarshal2, 3946
- tightUnmarshal, 3947
- TransactionIdMarshaller, 3945
- activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller, 3985
  - ~TransactionInfoMarshaller, 3986
  - createObject, 3986
  - getDataStructureType, 3987
  - looseMarshal, 3987
  - looseUnmarshal, 3987
  - tightMarshal1, 3988
  - tightMarshal2, 3988
  - tightUnmarshal, 3989
  - TransactionInfoMarshaller, 3986
- activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller, 4113
  - ~WireFormatInfoMarshaller, 4114
  - createObject, 4114
  - getDataStructureType, 4114
  - looseMarshal, 4114
  - looseUnmarshal, 4115
  - tightMarshal1, 4115
  - tightMarshal2, 4116
  - tightUnmarshal, 4116
  - WireFormatInfoMarshaller, 4114
- activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller, 4151
  - ~XATransactionIdMarshaller, 4152
  - createObject, 4152
  - getDataStructureType, 4153
  - looseMarshal, 4153
  - looseUnmarshal, 4153
  - tightMarshal1, 4154
  - tightMarshal2, 4154
  - tightUnmarshal, 4155
  - XATransactionIdMarshaller, 4152
- activemq::wireformat::openwire::marshal::v3, 113
- activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller, 189
  - ~ActiveMQBlobMessageMarshaller, 191
  - ActiveMQBlobMessageMarshaller, 191
  - createObject, 191
  - getDataStructureType, 191
  - looseMarshal, 191
  - looseUnmarshal, 192
  - tightMarshal1, 192
  - tightMarshal2, 193
  - tightUnmarshal, 193
- activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller, 235
  - ~ActiveMQBytesMessageMarshaller, 236
  - ActiveMQBytesMessageMarshaller, 236
  - createObject, 236
  - getDataStructureType, 236
  - looseMarshal, 236
  - looseUnmarshal, 237
  - tightMarshal1, 237
  - tightMarshal2, 238
  - tightUnmarshal, 238
- activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller, 324
  - ~ActiveMQDestinationMarshaller, 325
  - ActiveMQDestinationMarshaller, 325
  - looseMarshal, 325
  - looseUnmarshal, 325
  - tightMarshal1, 326
  - tightMarshal2, 326
  - tightUnmarshal, 327
- activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller, 364
  - ~ActiveMQMapMessageMarshaller, 366
  - ActiveMQMapMessageMarshaller, 366
  - createObject, 366
  - getDataStructureType, 366
  - looseMarshal, 366
  - looseUnmarshal, 367
  - tightMarshal1, 367
  - tightMarshal2, 368
  - tightUnmarshal, 368
- activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller, 393
  - ~ActiveMQMessageMarshaller, 394
  - ActiveMQMessageMarshaller, 394
  - createObject, 394
  - getDataStructureType, 394
  - looseMarshal, 394
  - looseUnmarshal, 395
  - tightMarshal1, 395
  - tightMarshal2, 396
  - tightUnmarshal, 396
- activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller, 440

- ~ActiveMQObjectMessageMarshaller, 442
- ActiveMQObjectMessageMarshaller, 442
- createObject, 442
- getDataStructureType, 442
- looseMarshal, 442
- looseUnmarshal, 443
- tightMarshal1, 443
- tightMarshal2, 444
- tightUnmarshal, 444
- activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller, 487
- ~ActiveMQQueueMarshaller, 488
- ActiveMQQueueMarshaller, 488
- createObject, 488
- getDataStructureType, 488
- looseMarshal, 488
- looseUnmarshal, 489
- tightMarshal1, 489
- tightMarshal2, 490
- tightUnmarshal, 490
- activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller, 553
- ~ActiveMQStreamMessageMarshaller, 554
- ActiveMQStreamMessageMarshaller, 554
- createObject, 554
- getDataStructureType, 554
- looseMarshal, 555
- looseUnmarshal, 555
- tightMarshal1, 555
- tightMarshal2, 556
- tightUnmarshal, 556
- activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller, 582
- ~ActiveMQTempDestinationMarshaller, 584
- ActiveMQTempDestinationMarshaller, 584
- looseMarshal, 584
- looseUnmarshal, 584
- tightMarshal1, 585
- tightMarshal2, 585
- tightUnmarshal, 586
- activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller, 611
- ~ActiveMQTempQueueMarshaller, 612
- ActiveMQTempQueueMarshaller, 612
- createObject, 612
- getDataStructureType, 612
- looseMarshal, 613
- looseUnmarshal, 613
- tightMarshal1, 614
- tightMarshal2, 614
- tightUnmarshal, 614
- activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller, 641
- ~ActiveMQTempTopicMarshaller, 642
- ActiveMQTempTopicMarshaller, 642
- createObject, 642
- getDataStructureType, 642
- looseMarshal, 643
- looseUnmarshal, 643
- tightMarshal1, 644
- tightMarshal2, 644
- tightUnmarshal, 644
- activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller, 671
- ~ActiveMQTextMessageMarshaller, 673
- ActiveMQTextMessageMarshaller, 673
- createObject, 673
- getDataStructureType, 673
- looseMarshal, 673
- looseUnmarshal, 674
- tightMarshal1, 674
- tightMarshal2, 675
- tightUnmarshal, 675
- activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller, 701
- ~ActiveMQTopicMarshaller, 702
- ActiveMQTopicMarshaller, 702
- createObject, 702
- getDataStructureType, 702
- looseMarshal, 703
- looseUnmarshal, 703
- tightMarshal1, 704
- tightMarshal2, 704
- tightUnmarshal, 705
- activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller, 771
- ~BaseCommandMarshaller, 772
- BaseCommandMarshaller, 772
- looseMarshal, 772
- looseUnmarshal, 772
- tightMarshal1, 775
- tightMarshal2, 776
- tightUnmarshal, 777

- activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller, 874
  - 878
  - ~BrokerIdMarshaller, 879
  - BrokerIdMarshaller, 879
  - createObject, 879
  - getDataStructureType, 879
  - looseMarshal, 880
  - looseUnmarshal, 880
  - tightMarshal1, 880
  - tightMarshal2, 881
  - tightUnmarshal, 881
- activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller, 910
  - ~BrokerInfoMarshaller, 912
  - BrokerInfoMarshaller, 912
  - createObject, 912
  - getDataStructureType, 912
  - looseMarshal, 912
  - looseUnmarshal, 913
  - tightMarshal1, 913
  - tightMarshal2, 914
  - tightUnmarshal, 914
- activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller, 1307
  - ~ConnectionControlMarshaller, 1308
  - ConnectionControlMarshaller, 1308
  - createObject, 1308
  - getDataStructureType, 1308
  - looseMarshal, 1308
  - looseUnmarshal, 1309
  - tightMarshal1, 1309
  - tightMarshal2, 1310
  - tightUnmarshal, 1310
- activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller, 1340
  - ~ConnectionErrorMarshaller, 1341
  - ConnectionErrorMarshaller, 1341
  - createObject, 1341
  - getDataStructureType, 1342
  - looseMarshal, 1342
  - looseUnmarshal, 1342
  - tightMarshal1, 1343
  - tightMarshal2, 1343
  - tightUnmarshal, 1344
- activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller, 1373
  - ~ConnectionIdMarshaller, 1374
  - ConnectionIdMarshaller, 1374
  - createObject, 1374
  - getDataStructureType, 1374
- activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller, 1374
  - looseUnmarshal, 1375
  - tightMarshal1, 1375
  - tightMarshal2, 1376
  - tightUnmarshal, 1376
- activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller, 1404
  - ~ConnectionInfoMarshaller, 1405
  - ConnectionInfoMarshaller, 1405
  - createObject, 1405
  - getDataStructureType, 1405
  - looseMarshal, 1406
  - looseUnmarshal, 1406
  - tightMarshal1, 1406
  - tightMarshal2, 1407
  - tightUnmarshal, 1407
- activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller, 1451
  - ~ConsumerControlMarshaller, 1452
  - ConsumerControlMarshaller, 1452
  - createObject, 1452
  - getDataStructureType, 1452
  - looseMarshal, 1453
  - looseUnmarshal, 1453
  - tightMarshal1, 1453
  - tightMarshal2, 1454
  - tightUnmarshal, 1454
- activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller, 1481
  - ~ConsumerIdMarshaller, 1482
  - ConsumerIdMarshaller, 1482
  - createObject, 1482
  - getDataStructureType, 1482
  - looseMarshal, 1483
  - looseUnmarshal, 1483
  - tightMarshal1, 1483
  - tightMarshal2, 1484
  - tightUnmarshal, 1484
- activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller, 1514
  - ~ConsumerInfoMarshaller, 1515
  - ConsumerInfoMarshaller, 1515
  - createObject, 1515
  - getDataStructureType, 1515
  - looseMarshal, 1516
  - looseUnmarshal, 1516
  - tightMarshal1, 1516
  - tightMarshal2, 1517
  - tightUnmarshal, 1517

- activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller, 1544
  - ~ControlCommandMarshaller, 1545
  - ControlCommandMarshaller, 1545
  - createObject, 1545
  - getDataStructureType, 1545
  - looseMarshal, 1545
  - looseUnmarshal, 1546
  - tightMarshal1, 1546
  - tightMarshal2, 1547
  - tightUnmarshal, 1547
- activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller, 1579
  - ~DataArrayResponseMarshaller, 1580
  - createObject, 1580
  - DataArrayResponseMarshaller, 1580
  - getDataStructureType, 1580
  - looseMarshal, 1580
  - looseUnmarshal, 1581
  - tightMarshal1, 1581
  - tightMarshal2, 1582
  - tightUnmarshal, 1582
- activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller, 1647
  - ~DataResponseMarshaller, 1649
  - createObject, 1649
  - DataResponseMarshaller, 1649
  - getDataStructureType, 1649
  - looseMarshal, 1649
  - looseUnmarshal, 1650
  - tightMarshal1, 1650
  - tightMarshal2, 1651
  - tightUnmarshal, 1651
- activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller, 1789
  - ~DestinationInfoMarshaller, 1790
  - createObject, 1790
  - DestinationInfoMarshaller, 1790
  - getDataStructureType, 1790
  - looseMarshal, 1790
  - looseUnmarshal, 1791
  - tightMarshal1, 1791
  - tightMarshal2, 1792
  - tightUnmarshal, 1792
- activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller, 1823
  - ~DiscoveryEventMarshaller, 1824
  - createObject, 1824
  - DiscoveryEventMarshaller, 1824
  - getDataStructureType, 1825
- activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, 2171
  - ~IntegerResponseMarshaller, 2172
  - createObject, 2172
  - getDataStructureType, 2172
  - IntegerResponseMarshaller, 2172
  - looseMarshal, 2173
  - looseUnmarshal, 2173
  - tightMarshal1, 2174
  - tightMarshal2, 2174
  - tightUnmarshal, 2175
- activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller, 2240
  - ~JournalQueueAckMarshaller, 2241
  - createObject, 2241
  - getDataStructureType, 2241
  - JournalQueueAckMarshaller, 2241
  - looseMarshal, 2241
  - looseUnmarshal, 2242
  - tightMarshal1, 2242
  - tightMarshal2, 2243
  - tightUnmarshal, 2243
- activemq::wireformat::openwire::marshal::v3::LooseMarshal, 1825
  - looseUnmarshal, 1825
  - tightMarshal1, 1826
  - tightMarshal2, 1826
  - tightUnmarshal, 1827
- activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller, 1906
  - ~ExceptionResponseMarshaller, 1907
  - createObject, 1907
  - ExceptionResponseMarshaller, 1907
  - getDataStructureType, 1908
  - looseMarshal, 1908
  - looseUnmarshal, 1908
  - tightMarshal1, 1909
  - tightMarshal2, 1909
  - tightUnmarshal, 1910
- activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller, 2010
  - ~FlushCommandMarshaller, 2011
  - createObject, 2011
  - FlushCommandMarshaller, 2011
  - getDataStructureType, 2012
  - DataResponseMarshaller, 2012
  - looseMarshal, 2012
  - looseUnmarshal, 2012
  - tightMarshal1, 2013
  - tightMarshal2, 2013
  - tightUnmarshal, 2014

- activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller, 2265
  - ~JournalTopicAckMarshaller, 2266
  - createObject, 2266
  - getDataStructureType, 2266
  - JournalTopicAckMarshaller, 2266
  - looseMarshal, 2266
  - looseUnmarshal, 2267
  - tightMarshal1, 2267
  - tightMarshal2, 2268
  - tightUnmarshal, 2268
- activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller, 2288
  - ~JournalTraceMarshaller, 2289
  - createObject, 2289
  - getDataStructureType, 2289
  - JournalTraceMarshaller, 2289
  - looseMarshal, 2289
  - looseUnmarshal, 2290
  - tightMarshal1, 2290
  - tightMarshal2, 2291
  - tightUnmarshal, 2291
- activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller, 2319
  - ~JournalTransactionMarshaller, 2320
  - createObject, 2320
  - getDataStructureType, 2321
  - JournalTransactionMarshaller, 2320
  - looseMarshal, 2321
  - looseUnmarshal, 2321
  - tightMarshal1, 2322
  - tightMarshal2, 2322
  - tightUnmarshal, 2323
- activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller, 2347
  - ~KeepAliveInfoMarshaller, 2348
  - createObject, 2348
  - getDataStructureType, 2348
  - KeepAliveInfoMarshaller, 2348
  - looseMarshal, 2348
  - looseUnmarshal, 2349
  - tightMarshal1, 2349
  - tightMarshal2, 2350
  - tightUnmarshal, 2350
- activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller, 2378
  - ~LastPartialCommandMarshaller, 2379
  - createObject, 2379
  - getDataStructureType, 2379
  - LastPartialCommandMarshaller, 2379
- activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller, 2433
  - ~LocalTransactionIdMarshaller, 2434
  - createObject, 2434
  - getDataStructureType, 2435
  - LocalTransactionIdMarshaller, 2434
- activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller, 2657
  - ~MarshallerFactory, 2568
  - configure, 2568
- activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller, 2695
  - ~MessageDispatchMarshaller, 2696
  - createObject, 2696
  - getDataStructureType, 2696
  - looseMarshal, 2696
  - looseUnmarshal, 2697
  - MessageDispatchMarshaller, 2696
  - tightMarshal1, 2697
  - tightMarshal2, 2698
  - tightUnmarshal, 2698
- activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller, 2729
  - ~MessageDispatchNotificationMarshaller, 2730
  - createObject, 2730
  - getDataStructureType, 2730
  - looseMarshal, 2731
  - looseUnmarshal, 2731

- MessageDispatchNotificationMarshaller, 2730
- tightMarshal1, 2731
- tightMarshal2, 2732
- tightUnmarshal, 2732
- activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller, 2767
  - ~MessageIdMarshaller, 2768
  - createObject, 2768
  - getDataStructureType, 2769
  - looseMarshal, 2769
  - looseUnmarshal, 2769
  - MessageIdMarshaller, 2768
  - tightMarshal1, 2770
  - tightMarshal2, 2770
  - tightUnmarshal, 2771
- activemq::wireformat::openwire::marshal::v3::MessageMarshaller, 2785
  - ~MessageMarshaller, 2786
  - looseMarshal, 2786
  - looseUnmarshal, 2786
  - MessageMarshaller, 2786
  - tightMarshal1, 2787
  - tightMarshal2, 2788
  - tightUnmarshal, 2788
- activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller, 2837
  - ~MessagePullMarshaller, 2838
  - createObject, 2838
  - getDataStructureType, 2838
  - looseMarshal, 2839
  - looseUnmarshal, 2839
  - MessagePullMarshaller, 2838
  - tightMarshal1, 2839
  - tightMarshal2, 2840
  - tightUnmarshal, 2840
- activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller, 2893
  - ~NetworkBridgeFilterMarshaller, 2894
  - createObject, 2894
  - getDataStructureType, 2894
  - looseMarshal, 2894
  - looseUnmarshal, 2895
  - NetworkBridgeFilterMarshaller, 2894
  - tightMarshal1, 2895
  - tightMarshal2, 2896
  - tightUnmarshal, 2896
- activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller, 3019
  - ~PartialCommandMarshaller, 3020
  - createObject, 3020
  - getDataStructureType, 3020
  - looseMarshal, 3021
  - looseUnmarshal, 3021
  - PartialCommandMarshaller, 3020
  - tightMarshal1, 3022
  - tightMarshal2, 3022
  - tightUnmarshal, 3023
- activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller, 3137
  - ~ProducerAckMarshaller, 3138
  - createObject, 3138
  - getDataStructureType, 3138
  - looseMarshal, 3139
  - looseUnmarshal, 3139
  - ProducerAckMarshaller, 3138
  - tightMarshal1, 3139
  - tightMarshal2, 3140
  - tightUnmarshal, 3140
- activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller, 3169
  - ~ProducerIdMarshaller, 3170
  - createObject, 3170
  - getDataStructureType, 3171
  - looseMarshal, 3171
  - looseUnmarshal, 3171
  - ProducerIdMarshaller, 3170
  - tightMarshal1, 3172
  - tightMarshal2, 3172
  - tightUnmarshal, 3173
- activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller, 3207
  - ~ProducerInfoMarshaller, 3208
  - createObject, 3208
  - getDataStructureType, 3209
  - looseMarshal, 3209
  - looseUnmarshal, 3209
  - ProducerInfoMarshaller, 3208
  - tightMarshal1, 3210
  - tightMarshal2, 3210
  - tightUnmarshal, 3211
- activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller, 3296
  - ~RemoveInfoMarshaller, 3297
  - createObject, 3297
  - getDataStructureType, 3298
  - looseMarshal, 3298
  - looseUnmarshal, 3298
  - RemoveInfoMarshaller, 3297
  - tightMarshal1, 3299



- tightMarshal2, 3299
- tightUnmarshal, 3300
- activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller, 3322
  - ~RemoveSubscriptionInfoMarshaller, 3323
- createObject, 3323
- getDataStructureType, 3324
- looseMarshal, 3324
- looseUnmarshal, 3324
- RemoveSubscriptionInfoMarshaller, 3323
- tightMarshal1, 3325
- tightMarshal2, 3325
- tightUnmarshal, 3326
- activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller, 3360
  - ~ReplayCommandMarshaller, 3361
- createObject, 3361
- getDataStructureType, 3361
- looseMarshal, 3361
- looseUnmarshal, 3362
- ReplayCommandMarshaller, 3361
- tightMarshal1, 3362
- tightMarshal2, 3363
- tightUnmarshal, 3363
- activemq::wireformat::openwire::marshal::v3::ResponseMarshaller, 3403
  - ~ResponseMarshaller, 3405
- createObject, 3405
- getDataStructureType, 3405
- looseMarshal, 3405
- looseUnmarshal, 3406
- ResponseMarshaller, 3405
- tightMarshal1, 3406
- tightMarshal2, 3407
- tightUnmarshal, 3408
- activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller, 3497
  - ~SessionIdMarshaller, 3498
- createObject, 3498
- getDataStructureType, 3498
- looseMarshal, 3498
- looseUnmarshal, 3499
- SessionIdMarshaller, 3498
- tightMarshal1, 3499
- tightMarshal2, 3500
- tightUnmarshal, 3500
- activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller, 3521
  - ~SessionInfoMarshaller, 3522
- createObject, 3522
- getDataStructureType, 3523
- looseMarshal, 3523
- looseUnmarshal, 3523
- SessionInfoMarshaller, 3522
- tightMarshal1, 3524
- tightMarshal2, 3524
- tightUnmarshal, 3525
- activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller, 3593
  - ~ShutdownInfoMarshaller, 3594
- createObject, 3594
- getDataStructureType, 3594
- looseMarshal, 3595
- looseUnmarshal, 3595
- ShutdownInfoMarshaller, 3594
- tightMarshal1, 3595
- tightMarshal2, 3596
- tightUnmarshal, 3596
- activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller, 3786
  - ~SubscriptionInfoMarshaller, 3788
- createObject, 3788
- getDataStructureType, 3788
- looseMarshal, 3788
- looseUnmarshal, 3789
- SubscriptionInfoMarshaller, 3788
- tightMarshal1, 3789
- tightMarshal2, 3789
- tightUnmarshal, 3790
- activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller, 3947
  - ~TransactionIdMarshaller, 3949
- looseMarshal, 3949
- looseUnmarshal, 3949
- tightMarshal1, 3950
- tightUnmarshal, 3950
- TransactionIdMarshaller, 3949
- activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller, 3972
  - ~TransactionInfoMarshaller, 3974
- createObject, 3974
- getDataStructureType, 3974
- looseMarshal, 3974
- looseUnmarshal, 3975
- tightMarshal1, 3975
- tightUnmarshal, 3976
- TransactionInfoMarshaller, 3974

- activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller, 333
  - 4125
  - ~WireFormatInfoMarshaller, 4126
  - createObject, 4126
  - getDataStructureType, 4126
  - looseMarshal, 4126
  - looseUnmarshal, 4127
  - tightMarshal1, 4127
  - tightMarshal2, 4128
  - tightUnmarshal, 4128
  - WireFormatInfoMarshaller, 4126
- activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller, 4164
  - ~XATransactionIdMarshaller, 4165
  - createObject, 4165
  - getDataStructureType, 4165
  - looseMarshal, 4166
  - looseUnmarshal, 4166
  - tightMarshal1, 4166
  - tightMarshal2, 4167
  - tightUnmarshal, 4167
  - XATransactionIdMarshaller, 4165
- activemq::wireformat::openwire::marshal::v4, 117
- activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller, 198
  - ~ActiveMQBlobMessageMarshaller, 199
  - ActiveMQBlobMessageMarshaller, 199
  - createObject, 199
  - getDataStructureType, 199
  - looseMarshal, 200
  - looseUnmarshal, 200
  - tightMarshal1, 200
  - tightMarshal2, 201
  - tightUnmarshal, 201
- activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller, 243
  - ~ActiveMQBytesMessageMarshaller, 244
  - ActiveMQBytesMessageMarshaller, 244
  - createObject, 244
  - getDataStructureType, 245
  - looseMarshal, 245
  - looseUnmarshal, 245
  - tightMarshal1, 246
  - tightMarshal2, 246
  - tightUnmarshal, 247
- activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationTypeMarshaller, 332
  - ~ActiveMQDestinationTypeMarshaller, 333
  - looseMarshal, 333
  - looseUnmarshal, 333
  - tightMarshal1, 334
  - tightMarshal2, 334
  - tightUnmarshal, 335
- activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller, 373
  - ~ActiveMQMapMessageMarshaller, 374
  - ActiveMQMapMessageMarshaller, 374
  - createObject, 374
  - getDataStructureType, 374
  - looseMarshal, 375
  - looseUnmarshal, 375
  - tightMarshal1, 375
  - tightMarshal2, 376
  - tightUnmarshal, 376
- activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller, 401
  - ~ActiveMQMessageMarshaller, 402
  - ActiveMQMessageMarshaller, 402
  - createObject, 402
  - getDataStructureType, 403
  - looseMarshal, 403
  - looseUnmarshal, 403
  - tightMarshal1, 404
  - tightMarshal2, 404
  - tightUnmarshal, 405
- activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller, 449
  - ~ActiveMQObjectMessageMarshaller, 450
  - ActiveMQObjectMessageMarshaller, 450
  - createObject, 450
  - getDataStructureType, 450
  - looseMarshal, 451
  - looseUnmarshal, 451
  - tightMarshal1, 451
  - tightMarshal2, 452
  - tightUnmarshal, 452
- activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller, 495
  - ~ActiveMQQueueMarshaller, 496
  - ActiveMQQueueMarshaller, 496
  - createObject, 496
  - getDataStructureType, 497
  - looseMarshal, 497

- looseUnmarshal, 497
- tightMarshal1, 498
- tightMarshal2, 498
- tightUnmarshal, 499
- activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller, 676
  - ~ActiveMQTextMessageMarshaller, 677
- activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller, 677
  - 561
  - ~ActiveMQStreamMessageMarshaller, 563
  - ActiveMQStreamMessageMarshaller, 563
  - createObject, 563
  - getDataStructureType, 563
  - looseMarshal, 563
  - looseUnmarshal, 564
  - tightMarshal1, 564
  - tightMarshal2, 565
  - tightUnmarshal, 565
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller, 707
  - 591
  - ~ActiveMQTempDestinationMarshaller, 592
  - ActiveMQTempDestinationMarshaller, 592
  - looseMarshal, 592
  - looseUnmarshal, 592
  - tightMarshal1, 593
  - tightMarshal2, 593
  - tightUnmarshal, 594
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller, 784
  - 619
  - ~ActiveMQTempQueueMarshaller, 621
  - ActiveMQTempQueueMarshaller, 621
  - createObject, 621
  - getDataStructureType, 621
  - looseMarshal, 621
  - looseUnmarshal, 622
  - tightMarshal1, 622
  - tightMarshal2, 623
  - tightUnmarshal, 623
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller, 645
  - ~ActiveMQTempTopicMarshaller, 646
  - ActiveMQTempTopicMarshaller, 646
  - createObject, 646
  - getDataStructureType, 647
  - looseMarshal, 647
  - looseUnmarshal, 647
  - tightMarshal1, 648
  - tightMarshal2, 648
  - tightUnmarshal, 649
- activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller, 677
  - createObject, 677
  - getDataStructureType, 677
  - looseMarshal, 677
  - looseUnmarshal, 678
  - tightMarshal1, 678
  - tightMarshal2, 679
  - tightUnmarshal, 679
- activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller, 705
  - ~ActiveMQTopicMarshaller, 707
  - ActiveMQTopicMarshaller, 707
  - createObject, 707
  - getDataStructureType, 707
  - looseMarshal, 707
  - looseUnmarshal, 708
  - tightMarshal1, 708
  - tightMarshal2, 709
  - tightUnmarshal, 709
- activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller, 778
  - ~BaseCommandMarshaller, 780
  - BaseCommandMarshaller, 780
  - looseMarshal, 780
  - looseUnmarshal, 780
  - tightMarshal1, 782
  - tightMarshal2, 783
  - tightUnmarshal, 784
- activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller, 882
  - ~BrokerIdMarshaller, 883
  - BrokerIdMarshaller, 883
  - createObject, 883
  - getDataStructureType, 883
  - looseMarshal, 884
  - looseUnmarshal, 884
  - tightMarshal1, 884
  - tightMarshal2, 885
  - tightUnmarshal, 885
- activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller, 915
  - ~BrokerInfoMarshaller, 916
  - BrokerInfoMarshaller, 916
  - createObject, 916
  - getDataStructureType, 916
  - looseMarshal, 916

- looseUnmarshal, 917
  - tightMarshal1, 917
  - tightMarshal2, 918
  - tightUnmarshal, 918
- activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller, 1311
  - ~ConsumerControlMarshaller, 1312
  - ConsumerControlMarshaller, 1312
  - createObject, 1312
  - getDataStructureType, 1312
  - looseMarshal, 1313
  - looseUnmarshal, 1313
  - tightMarshal1, 1313
  - tightMarshal2, 1314
  - tightUnmarshal, 1314
- activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller, 1344
  - ~ConsumerIdMarshaller, 1346
  - ConsumerIdMarshaller, 1346
  - createObject, 1346
  - getDataStructureType, 1346
  - looseMarshal, 1346
  - looseUnmarshal, 1347
  - tightMarshal1, 1347
  - tightMarshal2, 1348
  - tightUnmarshal, 1348
- activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller, 1377
  - ~ConsumerInfoMarshaller, 1378
  - ConsumerInfoMarshaller, 1378
  - createObject, 1378
  - getDataStructureType, 1378
  - looseMarshal, 1378
  - looseUnmarshal, 1379
  - tightMarshal1, 1379
  - tightMarshal2, 1380
  - tightUnmarshal, 1380
- activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller, 1408
  - ~ConsumerInfoMarshaller, 1409
  - ConsumerInfoMarshaller, 1409
  - createObject, 1409
  - getDataStructureType, 1410
  - looseMarshal, 1410
  - looseUnmarshal, 1410
  - tightMarshal1, 1411
  - tightMarshal2, 1411
  - tightUnmarshal, 1412
- activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller, 1455
  - ~ConsumerIdMarshaller, 1456
  - ConsumerIdMarshaller, 1456
  - createObject, 1456
  - getDataStructureType, 1457
  - looseMarshal, 1457
  - looseUnmarshal, 1457
  - tightMarshal1, 1458
  - tightMarshal2, 1458
  - tightUnmarshal, 1459
- activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller, 1485
  - ~ConsumerIdMarshaller, 1486
  - ConsumerIdMarshaller, 1486
  - createObject, 1486
  - getDataStructureType, 1487
  - looseMarshal, 1487
  - looseUnmarshal, 1487
  - tightMarshal1, 1488
  - tightMarshal2, 1488
  - tightUnmarshal, 1489
- activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller, 1518
  - ~ConsumerInfoMarshaller, 1519
  - ConsumerInfoMarshaller, 1519
  - createObject, 1519
  - getDataStructureType, 1520
  - looseMarshal, 1520
  - looseUnmarshal, 1520
  - tightMarshal1, 1521
  - tightMarshal2, 1521
  - tightUnmarshal, 1522
- activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller, 1548
  - ~ControlCommandMarshaller, 1549
  - ControlCommandMarshaller, 1549
  - createObject, 1549
  - getDataStructureType, 1549
  - looseMarshal, 1550
  - looseUnmarshal, 1550
  - tightMarshal1, 1550
  - tightMarshal2, 1551
  - tightUnmarshal, 1551
- activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller, 1583
  - ~DataArrayResponseMarshaller, 1584
  - createObject, 1584
  - DataArrayResponseMarshaller, 1584
  - getDataStructureType, 1584
  - looseMarshal, 1585
  - looseUnmarshal, 1585

- tightMarshal1, 1586
- tightMarshal2, 1586
- tightUnmarshal, 1587
- activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller, 1652
  - ~DataResponseMarshaller, 1653
  - createObject, 1653
  - DataResponseMarshaller, 1653
  - getDataStructureType, 1653
  - looseMarshal, 1653
  - looseUnmarshal, 1654
  - tightMarshal1, 1654
  - tightMarshal2, 1655
  - tightUnmarshal, 1655
- activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller, 1793
  - ~DestinationInfoMarshaller, 1794
  - createObject, 1794
  - DestinationInfoMarshaller, 1794
  - getDataStructureType, 1794
  - looseMarshal, 1795
  - looseUnmarshal, 1795
  - tightMarshal1, 1795
  - tightMarshal2, 1796
  - tightUnmarshal, 1796
- activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller, 1827
  - ~DiscoveryEventMarshaller, 1828
  - createObject, 1828
  - DiscoveryEventMarshaller, 1828
  - getDataStructureType, 1829
  - looseMarshal, 1829
  - looseUnmarshal, 1829
  - tightMarshal1, 1830
  - tightMarshal2, 1830
  - tightUnmarshal, 1831
- activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller, 1915
  - ~ExceptionResponseMarshaller, 1916
  - createObject, 1916
  - ExceptionResponseMarshaller, 1916
  - getDataStructureType, 1916
  - looseMarshal, 1916
  - looseUnmarshal, 1917
  - tightMarshal1, 1917
  - tightMarshal2, 1918
  - tightUnmarshal, 1918
- activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller, 2014
  - ~FlushCommandMarshaller, 2016
  - createObject, 2016
  - FlushCommandMarshaller, 2016
  - getDataStructureType, 2016
  - looseMarshal, 2016
  - looseUnmarshal, 2017
  - tightMarshal1, 2017
  - tightMarshal2, 2018
  - tightUnmarshal, 2018
- activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller, 2175
  - ~IntegerResponseMarshaller, 2177
  - createObject, 2177
  - getDataStructureType, 2177
  - IntegerResponseMarshaller, 2177
  - looseUnmarshal, 2178
  - tightMarshal1, 2178
  - tightMarshal2, 2179
  - tightUnmarshal, 2179
- activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller, 2244
  - ~JournalQueueAckMarshaller, 2245
  - createObject, 2245
  - getDataStructureType, 2245
  - JournalQueueAckMarshaller, 2245
  - looseMarshal, 2245
  - looseUnmarshal, 2246
  - tightMarshal1, 2246
  - tightMarshal2, 2247
  - tightUnmarshal, 2247
- activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller, 2273
  - ~JournalTopicAckMarshaller, 2274
  - createObject, 2274
  - getDataStructureType, 2274
  - JournalTopicAckMarshaller, 2274
  - looseMarshal, 2274
  - looseUnmarshal, 2275
  - tightMarshal1, 2275
  - tightMarshal2, 2276
  - tightUnmarshal, 2276
- activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller, 2296
  - ~JournalTraceMarshaller, 2297
  - createObject, 2297
  - getDataStructureType, 2297
  - JournalTraceMarshaller, 2297
  - looseMarshal, 2297
  - looseUnmarshal, 2298
  - tightMarshal1, 2298

- tightMarshal2, 2299
- tightUnmarshal, 2299
- activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller, 2327
  - ~JournalTransactionMarshaller, 2328
  - createObject, 2328
  - getDataStructureType, 2329
  - JournalTransactionMarshaller, 2328
  - looseMarshal, 2329
  - looseUnmarshal, 2329
  - tightMarshal1, 2330
  - tightMarshal2, 2330
  - tightUnmarshal, 2331
- activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller, 2351
  - ~KeepAliveInfoMarshaller, 2352
  - createObject, 2352
  - getDataStructureType, 2352
  - KeepAliveInfoMarshaller, 2352
  - looseMarshal, 2353
  - looseUnmarshal, 2353
  - tightMarshal1, 2353
  - tightMarshal2, 2354
  - tightUnmarshal, 2354
- activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller, 2391
  - ~LastPartialCommandMarshaller, 2392
  - createObject, 2392
  - getDataStructureType, 2392
  - LastPartialCommandMarshaller, 2392
  - looseMarshal, 2392
  - looseUnmarshal, 2393
  - tightMarshal1, 2393
  - tightMarshal2, 2394
  - tightUnmarshal, 2394
- activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller, 2442
  - ~LocalTransactionIdMarshaller, 2443
  - createObject, 2443
  - getDataStructureType, 2443
  - LocalTransactionIdMarshaller, 2443
  - looseMarshal, 2443
  - looseUnmarshal, 2444
  - tightMarshal1, 2444
  - tightMarshal2, 2445
  - tightUnmarshal, 2445
- activemq::wireformat::openwire::marshal::v4::MarshallerFactory, 2568
  - ~MarshallerFactory, 2569
  - configure, 2569
- activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller, 2661
  - createObject, 2662
  - getDataStructureType, 2663
  - looseMarshal, 2663
  - looseUnmarshal, 2663
  - MessageAckMarshaller, 2662
  - tightMarshal1, 2664
  - tightMarshal2, 2664
  - tightUnmarshal, 2665
- activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller, 2703
  - createObject, 2704
  - getDataStructureType, 2705
  - looseMarshal, 2705
  - looseUnmarshal, 2705
  - MessageDispatchMarshaller, 2704
  - tightMarshal1, 2706
  - tightMarshal2, 2706
  - tightUnmarshal, 2707
- activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller, 2733
  - createObject, 2734
  - getDataStructureType, 2735
  - looseMarshal, 2735
  - looseUnmarshal, 2735
  - MessageDispatchNotificationMarshaller, 2734
  - tightMarshal1, 2736
  - tightMarshal2, 2736
  - tightUnmarshal, 2737
- activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller, 2759
  - ~MessageIdMarshaller, 2760
  - createObject, 2760
  - getDataStructureType, 2761
  - looseMarshal, 2761
  - looseUnmarshal, 2761
  - MessageIdMarshaller, 2760
  - tightMarshal1, 2762
  - tightMarshal2, 2762
  - tightUnmarshal, 2763
- activemq::wireformat::openwire::marshal::v4::MessageMarshaller, 2793
  - ~MessageMarshaller, 2795
  - looseMarshal, 2795

- looseUnmarshal, 2795
  - MessageMarshaller, 2795
  - tightMarshal1, 2796
  - tightMarshal2, 2797
  - tightUnmarshal, 2797
- activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller, 2841
  - ~MessagePullMarshaller, 2842
  - createObject, 2842
  - getDataStructureType, 2843
  - looseMarshal, 2843
  - looseUnmarshal, 2843
  - MessagePullMarshaller, 2842
  - tightMarshal1, 2844
  - tightMarshal2, 2844
  - tightUnmarshal, 2845
- activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller, 2897
  - ~NetworkBridgeFilterMarshaller, 2898
  - createObject, 2898
  - getDataStructureType, 2898
  - looseMarshal, 2898
  - looseUnmarshal, 2899
  - NetworkBridgeFilterMarshaller, 2898
  - tightMarshal1, 2899
  - tightMarshal2, 2900
  - tightUnmarshal, 2900
- activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller, 3023
  - ~PartialCommandMarshaller, 3025
  - createObject, 3025
  - getDataStructureType, 3025
  - looseMarshal, 3025
  - looseUnmarshal, 3026
  - PartialCommandMarshaller, 3025
  - tightMarshal1, 3026
  - tightMarshal2, 3027
  - tightUnmarshal, 3027
- activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller, 3133
  - ~ProducerAckMarshaller, 3134
  - createObject, 3134
  - getDataStructureType, 3134
  - looseMarshal, 3134
  - looseUnmarshal, 3135
  - ProducerAckMarshaller, 3134
  - tightMarshal1, 3135
  - tightMarshal2, 3136
  - tightUnmarshal, 3136
- activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller, 3165
  - ~ProducerIdMarshaller, 3166
  - createObject, 3166
  - getDataStructureType, 3167
  - looseMarshal, 3167
  - looseUnmarshal, 3167
  - ProducerIdMarshaller, 3166
  - tightMarshal1, 3168
  - tightMarshal2, 3168
  - tightUnmarshal, 3169
- activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller, 3190
  - ~ProducerInfoMarshaller, 3191
  - createObject, 3191
  - getDataStructureType, 3192
  - looseMarshal, 3192
  - looseUnmarshal, 3192
  - ProducerInfoMarshaller, 3191
  - tightMarshal1, 3193
  - tightMarshal2, 3193
  - tightUnmarshal, 3194
- activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller, 3309
  - ~RemoveInfoMarshaller, 3310
  - createObject, 3310
  - getDataStructureType, 3310
  - looseMarshal, 3311
  - looseUnmarshal, 3311
  - RemoveInfoMarshaller, 3310
  - tightMarshal1, 3311
  - tightMarshal2, 3312
  - tightUnmarshal, 3312
- activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller, 3339
  - ~RemoveSubscriptionInfoMarshaller, 3340
  - createObject, 3340
  - getDataStructureType, 3341
  - looseMarshal, 3341
  - looseUnmarshal, 3341
  - RemoveSubscriptionInfoMarshaller, 3340
  - tightMarshal1, 3342
  - tightMarshal2, 3342
  - tightUnmarshal, 3343
- activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller, 3347
  - ~ReplayCommandMarshaller, 3348
  - createObject, 3348
  - getDataStructureType, 3348

- looseMarshal, 3349
- looseUnmarshal, 3349
- ReplayCommandMarshaller, 3348
- tightMarshal1, 3349
- tightMarshal2, 3350
- tightUnmarshal, 3350
- activemq::wireformat::openwire::marshal::v4::ResponseIdMarshaller, 3388
  - ~ResponseMarshaller, 3390
  - createObject, 3390
  - getDataStructureType, 3390
  - looseMarshal, 3390
  - looseUnmarshal, 3391
  - ResponseMarshaller, 3390
  - tightMarshal1, 3391
  - tightMarshal2, 3392
  - tightUnmarshal, 3393
- activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller, 3485
  - ~SessionIdMarshaller, 3486
  - createObject, 3486
  - getDataStructureType, 3486
  - looseMarshal, 3486
  - looseUnmarshal, 3487
  - SessionIdMarshaller, 3486
  - tightMarshal1, 3487
  - tightMarshal2, 3488
  - tightUnmarshal, 3488
- activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller, 3530
  - ~SessionInfoMarshaller, 3531
  - createObject, 3531
  - getDataStructureType, 3531
  - looseMarshal, 3531
  - looseUnmarshal, 3532
  - SessionInfoMarshaller, 3531
  - tightMarshal1, 3532
  - tightMarshal2, 3533
  - tightUnmarshal, 3533
- activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller, 3597
  - ~ShutdownInfoMarshaller, 3598
  - createObject, 3598
  - getDataStructureType, 3599
  - looseMarshal, 3599
  - looseUnmarshal, 3599
  - ShutdownInfoMarshaller, 3598
  - tightMarshal1, 3600
  - tightMarshal2, 3600
  - tightUnmarshal, 3601
- activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller, 3799
  - ~SubscriptionInfoMarshaller, 3800
  - createObject, 3800
  - getDataStructureType, 3800
  - looseMarshal, 3800
  - ResponseIdMarshaller, 3801
  - SubscriptionInfoMarshaller, 3800
  - tightMarshal1, 3801
  - tightMarshal2, 3802
  - tightUnmarshal, 3802
- activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller, 3951
  - ~TransactionIdMarshaller, 3953
  - looseMarshal, 3953
  - looseUnmarshal, 3953
  - tightMarshal1, 3954
  - tightMarshal2, 3954
  - tightUnmarshal, 3955
  - TransactionIdMarshaller, 3953
- activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller, 3981
  - ~TransactionInfoMarshaller, 3982
  - createObject, 3982
  - getDataStructureType, 3982
  - looseMarshal, 3983
  - looseUnmarshal, 3983
  - tightMarshal1, 3983
  - tightMarshal2, 3984
  - tightUnmarshal, 3984
  - TransactionInfoMarshaller, 3982
- activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller, 4117
  - ~WireFormatInfoMarshaller, 4118
  - createObject, 4118
  - getDataStructureType, 4118
  - looseMarshal, 4118
  - looseUnmarshal, 4119
  - tightMarshal1, 4119
  - tightMarshal2, 4120
  - tightUnmarshal, 4120
  - WireFormatInfoMarshaller, 4118
- activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller, 4155
  - ~XATransactionIdMarshaller, 4157
  - createObject, 4157
  - getDataStructureType, 4157
  - looseMarshal, 4157
  - looseUnmarshal, 4158
  - tightMarshal1, 4158



- tightMarshal2, 4159
- tightUnmarshal, 4159
- XATransactionIdMarshaller, 4157
- activemq::wireformat::openwire::marshal::v5, 121
- activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller, 206
  - ~ActiveMQBlobMessageMarshaller, 208
  - ActiveMQBlobMessageMarshaller, 208
  - createObject, 208
  - getDataStructureType, 208
  - looseMarshal, 208
  - looseUnmarshal, 209
  - tightMarshal1, 209
  - tightMarshal2, 210
  - tightUnmarshal, 210
- activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller, 247
  - ~ActiveMQBytesMessageMarshaller, 249
  - ActiveMQBytesMessageMarshaller, 249
  - createObject, 249
  - getDataStructureType, 249
  - looseMarshal, 249
  - looseUnmarshal, 250
  - tightMarshal1, 250
  - tightMarshal2, 251
  - tightUnmarshal, 251
- activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller, 336
  - ~ActiveMQDestinationMarshaller, 337
  - ActiveMQDestinationMarshaller, 337
  - looseMarshal, 337
  - looseUnmarshal, 337
  - tightMarshal1, 338
  - tightMarshal2, 338
  - tightUnmarshal, 339
- activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller, 377
  - ~ActiveMQMapMessageMarshaller, 378
  - ActiveMQMapMessageMarshaller, 378
  - createObject, 378
  - getDataStructureType, 379
  - looseMarshal, 379
  - looseUnmarshal, 379
  - tightMarshal1, 380
  - tightMarshal2, 380
  - tightUnmarshal, 381
- activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller, 405
  - ~ActiveMQMessageMarshaller, 407
  - ActiveMQMessageMarshaller, 407
  - createObject, 407
  - getDataStructureType, 407
  - looseMarshal, 407
  - looseUnmarshal, 408
  - tightMarshal1, 408
  - tightMarshal2, 409
  - tightUnmarshal, 409
- activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller, 453
  - ~ActiveMQObjectMessageMarshaller, 454
  - ActiveMQObjectMessageMarshaller, 454
- activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller, 499
  - ~ActiveMQQueueMarshaller, 501
  - ActiveMQQueueMarshaller, 501
  - createObject, 501
  - getDataStructureType, 501
  - looseMarshal, 501
  - looseUnmarshal, 502
  - tightMarshal1, 502
  - tightMarshal2, 503
  - tightUnmarshal, 503
- activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller, 566
  - ~ActiveMQStreamMessageMarshaller, 567
  - ActiveMQStreamMessageMarshaller, 567
  - createObject, 567
  - getDataStructureType, 567
  - looseMarshal, 567
  - looseUnmarshal, 568
  - tightMarshal1, 568
  - tightMarshal2, 569
  - tightUnmarshal, 569
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller, 595

- ~ActiveMQTempDestinationMarshaller, 596
- ActiveMQTempDestinationMarshaller, 596
- looseMarshal, 596
- looseUnmarshal, 596
- tightMarshal1, 597
- tightMarshal2, 597
- tightUnmarshal, 598
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller, 624
  - ~ActiveMQTempQueueMarshaller, 625
  - ActiveMQTempQueueMarshaller, 625
  - createObject, 625
  - getDataStructureType, 625
  - looseMarshal, 625
  - looseUnmarshal, 626
  - tightMarshal1, 626
  - tightMarshal2, 627
  - tightUnmarshal, 627
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller, 654
  - ~ActiveMQTempTopicMarshaller, 655
  - ActiveMQTempTopicMarshaller, 655
  - createObject, 655
  - getDataStructureType, 655
  - looseMarshal, 655
  - looseUnmarshal, 656
  - tightMarshal1, 656
  - tightMarshal2, 657
  - tightUnmarshal, 657
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller, 684
  - ~ActiveMQTextMessageMarshaller, 685
  - ActiveMQTextMessageMarshaller, 685
  - createObject, 685
  - getDataStructureType, 686
  - looseMarshal, 686
  - looseUnmarshal, 686
  - tightMarshal1, 687
  - tightMarshal2, 687
  - tightUnmarshal, 688
- activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller, 714
  - ~ActiveMQTopicMarshaller, 715
  - ActiveMQTopicMarshaller, 715
  - createObject, 715
  - getDataStructureType, 715
  - looseMarshal, 716
  - looseUnmarshal, 716
  - tightMarshal1, 716
  - tightMarshal2, 717
  - tightUnmarshal, 717
- activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller, 793
  - ~BaseCommandMarshaller, 794
  - BaseCommandMarshaller, 794
  - looseMarshal, 794
  - looseUnmarshal, 795
  - tightMarshal1, 796
  - tightMarshal2, 797
  - tightUnmarshal, 798
- activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, 890
  - ~BrokerIdMarshaller, 891
  - BrokerIdMarshaller, 891
  - createObject, 891
  - getDataStructureType, 892
  - looseMarshal, 892
  - looseUnmarshal, 892
  - tightMarshal1, 893
  - tightMarshal2, 893
  - tightUnmarshal, 894
- activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller, 923
  - ~BrokerInfoMarshaller, 924
  - BrokerInfoMarshaller, 924
  - createObject, 924
  - getDataStructureType, 925
  - looseMarshal, 925
  - looseUnmarshal, 925
  - tightMarshal1, 926
  - tightMarshal2, 926
  - tightUnmarshal, 927
- activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, 1319
  - ~ConnectionControlMarshaller, 1321
  - ConnectionControlMarshaller, 1321
  - createObject, 1321
  - getDataStructureType, 1321
  - looseMarshal, 1321
  - looseUnmarshal, 1322
  - tightMarshal1, 1322
  - tightMarshal2, 1323
  - tightUnmarshal, 1323
- activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller, 1353
  - ~ConnectionErrorMarshaller, 1354
  - ConnectionErrorMarshaller, 1354

- createObject, 1354
- getDataSetType, 1354
- looseMarshal, 1355
- looseUnmarshal, 1355
- tightMarshal1, 1355
- tightMarshal2, 1356
- tightUnmarshal, 1356
- activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, 1385
  - ~ConsumerInfoMarshaller, 1386
  - ConsumerInfoMarshaller, 1386
  - createObject, 1386
  - getDataSetType, 1386
  - looseMarshal, 1386
  - looseUnmarshal, 1387
  - tightMarshal1, 1387
  - tightMarshal2, 1388
  - tightUnmarshal, 1388
- activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller, 1417
  - ~ControlCommandMarshaller, 1418
  - ControlCommandMarshaller, 1418
  - createObject, 1418
  - getDataSetType, 1418
  - looseMarshal, 1418
  - looseUnmarshal, 1419
  - tightMarshal1, 1419
  - tightMarshal2, 1420
  - tightUnmarshal, 1420
- activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller, 1464
  - ~ConsumerControlMarshaller, 1465
  - ConsumerControlMarshaller, 1465
  - createObject, 1465
  - getDataSetType, 1465
  - looseMarshal, 1465
  - looseUnmarshal, 1466
  - tightMarshal1, 1466
  - tightMarshal2, 1467
  - tightUnmarshal, 1467
- activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller, 1493
  - ~ConsumerIdMarshaller, 1494
  - ConsumerIdMarshaller, 1494
  - createObject, 1494
  - getDataSetType, 1495
  - looseMarshal, 1495
  - looseUnmarshal, 1495
  - tightMarshal1, 1496
  - tightMarshal2, 1496
- tightUnmarshal, 1497
- activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, 1527
  - ~ConsumerInfoMarshaller, 1528
  - ConsumerInfoMarshaller, 1528
  - createObject, 1528
  - getDataSetType, 1528
  - looseMarshal, 1528
  - looseUnmarshal, 1529
  - tightMarshal1, 1529
  - tightMarshal2, 1530
  - tightUnmarshal, 1530
- activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller, 1556
  - ~ControlCommandMarshaller, 1558
  - ControlCommandMarshaller, 1558
  - createObject, 1558
  - getDataSetType, 1558
  - looseMarshal, 1558
  - looseUnmarshal, 1559
  - tightMarshal1, 1559
  - tightMarshal2, 1560
  - tightUnmarshal, 1560
- activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller, 1592
  - ~DataArrayResponseMarshaller, 1593
  - createObject, 1593
  - DataArrayResponseMarshaller, 1593
  - getDataSetType, 1593
  - looseMarshal, 1593
  - looseUnmarshal, 1594
  - tightMarshal1, 1594
  - tightMarshal2, 1595
  - tightUnmarshal, 1595
- activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller, 1635
  - ~DataResponseMarshaller, 1636
  - createObject, 1636
  - DataResponseMarshaller, 1636
  - getDataSetType, 1636
  - looseMarshal, 1636
  - looseUnmarshal, 1637
  - tightMarshal1, 1637
  - tightMarshal2, 1638
  - tightUnmarshal, 1638
- activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller, 1806
  - ~DestinationInfoMarshaller, 1807
  - createObject, 1807
  - DestinationInfoMarshaller, 1807

- getDataStructureType, 1807
  - looseMarshal, 1807
  - looseUnmarshal, 1808
  - tightMarshal1, 1808
  - tightMarshal2, 1809
  - tightUnmarshal, 1809
- activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller, 1835
  - ~DiscoveryEventMarshaller, 1836
  - createObject, 1836
  - DiscoveryEventMarshaller, 1836
  - getDataStructureType, 1837
  - looseMarshal, 1837
  - looseUnmarshal, 1837
  - tightMarshal1, 1838
  - tightMarshal2, 1838
  - tightUnmarshal, 1839
- activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller, 1910
  - ~ExceptionResponseMarshaller, 1912
  - createObject, 1912
  - ExceptionResponseMarshaller, 1912
  - getDataStructureType, 1912
  - looseMarshal, 1912
  - looseUnmarshal, 1913
  - tightMarshal1, 1913
  - tightMarshal2, 1914
  - tightUnmarshal, 1914
- activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller, 2023
  - ~FlushCommandMarshaller, 2024
  - createObject, 2024
  - FlushCommandMarshaller, 2024
  - getDataStructureType, 2024
  - looseMarshal, 2025
  - looseUnmarshal, 2025
  - tightMarshal1, 2025
  - tightMarshal2, 2026
  - tightUnmarshal, 2026
- activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller, 2184
  - ~IntegerResponseMarshaller, 2185
  - createObject, 2185
  - getDataStructureType, 2185
  - IntegerResponseMarshaller, 2185
  - looseMarshal, 2186
  - looseUnmarshal, 2186
  - tightMarshal1, 2187
  - tightMarshal2, 2187
  - tightUnmarshal, 2188
- activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller, 2236
  - ~JournalQueueAckMarshaller, 2237
  - createObject, 2237
  - getDataStructureType, 2237
  - JournalQueueAckMarshaller, 2237
  - looseMarshal, 2237
  - looseUnmarshal, 2238
  - tightMarshal1, 2238
  - tightMarshal2, 2239
  - tightUnmarshal, 2239
- activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller, 2256
  - ~JournalTopicAckMarshaller, 2258
  - createObject, 2258
  - getDataStructureType, 2258
  - JournalTopicAckMarshaller, 2258
  - looseMarshal, 2258
  - looseUnmarshal, 2259
  - tightMarshal1, 2259
  - tightMarshal2, 2259
  - tightUnmarshal, 2260
- activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller, 2304
  - ~JournalTraceMarshaller, 2305
  - createObject, 2305
  - getDataStructureType, 2305
  - JournalTraceMarshaller, 2305
  - looseMarshal, 2305
  - looseUnmarshal, 2306
  - tightMarshal1, 2306
  - tightMarshal2, 2307
  - tightUnmarshal, 2307
- activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller, 2323
  - ~JournalTransactionMarshaller, 2324
  - createObject, 2324
  - getDataStructureType, 2325
  - JournalTransactionMarshaller, 2324
  - looseMarshal, 2325
  - looseUnmarshal, 2325
  - tightMarshal1, 2326
  - tightMarshal2, 2326
  - tightUnmarshal, 2327
- activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller, 2355
  - ~KeepAliveInfoMarshaller, 2356
  - createObject, 2356
  - getDataStructureType, 2357
  - KeepAliveInfoMarshaller, 2356

- looseMarshal, 2357
- looseUnmarshal, 2357
- tightMarshal1, 2358
- tightMarshal2, 2358
- tightUnmarshal, 2359
- activemq::wireformat::openwire::marshal::v5::LastMessageDispatchNotificationMarshaller, 2386
  - ~LastPartialCommandMarshaller, 2388
  - createObject, 2388
  - getDataStructureType, 2388
  - LastPartialCommandMarshaller, 2388
  - looseMarshal, 2388
  - looseUnmarshal, 2389
  - tightMarshal1, 2389
  - tightMarshal2, 2390
  - tightUnmarshal, 2390
- activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller, 2437
  - ~LocalTransactionIdMarshaller, 2439
  - createObject, 2439
  - getDataStructureType, 2439
  - LocalTransactionIdMarshaller, 2439
  - looseMarshal, 2439
  - looseUnmarshal, 2440
  - tightMarshal1, 2440
  - tightMarshal2, 2441
  - tightUnmarshal, 2441
- activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller, 2569
  - ~MessageIdMarshaller, 2569
  - configure, 2569
- activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller, 2670
  - ~MessageAckMarshaller, 2671
  - createObject, 2671
  - getDataStructureType, 2671
  - looseMarshal, 2671
  - looseUnmarshal, 2672
  - MessageAckMarshaller, 2671
  - tightMarshal1, 2672
  - tightMarshal2, 2673
  - tightUnmarshal, 2673
- activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller, 2699
  - ~MessageDispatchMarshaller, 2700
  - createObject, 2700
  - getDataStructureType, 2700
  - looseMarshal, 2701
  - looseUnmarshal, 2701
  - MessageDispatchMarshaller, 2700
- tightMarshal1, 2701
- tightMarshal2, 2702
- tightUnmarshal, 2702
- activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller, 2742
  - createObject, 2743
  - getDataStructureType, 2743
  - looseMarshal, 2743
  - looseUnmarshal, 2744
  - MessageDispatchNotificationMarshaller, 2743
  - tightMarshal1, 2744
  - tightMarshal2, 2745
  - tightUnmarshal, 2745
- activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller, 2763
  - ~MessageIdMarshaller, 2764
  - createObject, 2764
  - getDataStructureType, 2765
  - looseMarshal, 2765
  - looseUnmarshal, 2765
  - MessageIdMarshaller, 2764
  - tightMarshal1, 2766
  - tightMarshal2, 2766
  - tightUnmarshal, 2767
- activemq::wireformat::openwire::marshal::v5::MessageMarshaller, 2780
  - ~MessageMarshaller, 2781
  - looseMarshal, 2781
  - MessageMarshaller, 2781
  - tightMarshal1, 2783
  - tightMarshal2, 2783
  - tightUnmarshal, 2784
- activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller, 2833
  - ~MessagePullMarshaller, 2834
  - createObject, 2834
  - getDataStructureType, 2834
  - looseMarshal, 2834
  - MessagePullMarshaller, 2834
  - tightMarshal1, 2835
  - tightMarshal2, 2836
  - tightUnmarshal, 2836
- activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller, 2889
  - ~NetworkBridgeFilterMarshaller, 2890

- createObject, 2890
- getDataStructureType, 2890
- looseMarshal, 2890
- looseUnmarshal, 2891
- NetworkBridgeFilterMarshaller, 2890
- tightMarshal1, 2891
- tightMarshal2, 2892
- tightUnmarshal, 2892
- activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, 3014
  - ~PartialCommandMarshaller, 3016
  - createObject, 3016
  - getDataStructureType, 3016
  - looseMarshal, 3016
  - looseUnmarshal, 3017
  - PartialCommandMarshaller, 3016
  - tightMarshal1, 3017
  - tightMarshal2, 3018
  - tightUnmarshal, 3018
- activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller, 3141
  - ~ProducerAckMarshaller, 3142
  - createObject, 3142
  - getDataStructureType, 3143
  - looseMarshal, 3143
  - looseUnmarshal, 3143
  - ProducerAckMarshaller, 3142
  - tightMarshal1, 3144
  - tightMarshal2, 3144
  - tightUnmarshal, 3145
- activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller, 3173
  - ~ProducerIdMarshaller, 3174
  - createObject, 3174
  - getDataStructureType, 3175
  - looseMarshal, 3175
  - looseUnmarshal, 3175
  - ProducerIdMarshaller, 3174
  - tightMarshal1, 3176
  - tightMarshal2, 3176
  - tightUnmarshal, 3177
- activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller, 3203
  - ~ProducerInfoMarshaller, 3204
  - createObject, 3204
  - getDataStructureType, 3204
  - looseMarshal, 3205
  - looseUnmarshal, 3205
  - ProducerInfoMarshaller, 3204
  - tightMarshal1, 3205
- tightMarshal2, 3206
- tightUnmarshal, 3206
- activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller, 3305
  - ~RemoveInfoMarshaller, 3306
  - createObject, 3306
  - getDataStructureType, 3306
  - looseMarshal, 3306
  - PartialCommandMarshaller, 3307
  - RemoveInfoMarshaller, 3306
  - tightMarshal1, 3307
  - tightMarshal2, 3308
  - tightUnmarshal, 3308
- activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller, 3335
  - ~RemoveSubscriptionInfoMarshaller, 3336
  - createObject, 3336
  - getDataStructureType, 3336
  - looseUnmarshal, 3337
  - RemoveSubscriptionInfoMarshaller, 3336
  - tightMarshal1, 3337
  - tightMarshal2, 3338
  - tightUnmarshal, 3338
- activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller, 3368
  - ~ReplayCommandMarshaller, 3369
  - createObject, 3369
  - getDataStructureType, 3370
  - looseUnmarshal, 3370
  - ReplayCommandMarshaller, 3369
  - tightMarshal1, 3371
  - tightMarshal2, 3371
  - tightUnmarshal, 3372
- activemq::wireformat::openwire::marshal::v5::ResponseMarshaller, 3398
  - ~ResponseMarshaller, 3400
  - createObject, 3400
  - getDataStructureType, 3400
  - looseUnmarshal, 3401
  - ResponseMarshaller, 3400
  - tightMarshal1, 3401
  - tightMarshal2, 3402
  - tightUnmarshal, 3403
- activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller, 3493
  - ~SessionIdMarshaller, 3494

- createObject, 3494
- getDataSetType, 3494
- looseMarshal, 3494
- looseUnmarshal, 3495
- SessionIdMarshaller, 3494
- tightMarshal1, 3495
- tightMarshal2, 3496
- tightUnmarshal, 3496
- activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller, 3513
  - ~SessionInfoMarshaller, 3514
  - createObject, 3514
  - getDataSetType, 3514
  - looseMarshal, 3514
  - looseUnmarshal, 3515
  - SessionInfoMarshaller, 3514
  - tightMarshal1, 3515
  - tightMarshal2, 3516
  - tightUnmarshal, 3516
- activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller, 3589
  - ~ShutdownInfoMarshaller, 3590
  - createObject, 3590
  - getDataSetType, 3590
  - looseMarshal, 3590
  - looseUnmarshal, 3591
  - ShutdownInfoMarshaller, 3590
  - tightMarshal1, 3591
  - tightMarshal2, 3592
  - tightUnmarshal, 3592
- activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller, 3795
  - ~SubscriptionInfoMarshaller, 3796
  - createObject, 3796
  - getDataSetType, 3796
  - looseMarshal, 3796
  - looseUnmarshal, 3797
  - SubscriptionInfoMarshaller, 3796
  - tightMarshal1, 3797
  - tightMarshal2, 3798
  - tightUnmarshal, 3798
- activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller, 3935
  - ~TransactionIdMarshaller, 3937
  - looseMarshal, 3937
  - looseUnmarshal, 3937
  - tightMarshal1, 3938
  - tightMarshal2, 3938
  - tightUnmarshal, 3939
  - TransactionIdMarshaller, 3937
- activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller, 3964
  - ~TransactionInfoMarshaller, 3965
  - createObject, 3965
  - getDataSetType, 3965
  - looseMarshal, 3966
  - looseUnmarshal, 3966
  - tightMarshal1, 3967
  - tightMarshal2, 3967
  - tightUnmarshal, 3967
  - TransactionInfoMarshaller, 3965
- activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller, 4104
  - ~WireFormatInfoMarshaller, 4106
  - createObject, 4106
  - getDataSetType, 4106
  - looseMarshal, 4106
  - looseUnmarshal, 4107
  - tightMarshal1, 4107
  - tightMarshal2, 4107
  - tightUnmarshal, 4108
  - WireFormatInfoMarshaller, 4106
- activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller, 4168
  - ~XATransactionIdMarshaller, 4169
  - createObject, 4169
  - getDataSetType, 4170
  - looseMarshal, 4170
  - looseUnmarshal, 4170
  - tightMarshal1, 4171
  - tightMarshal2, 4171
  - tightUnmarshal, 4172
  - XATransactionIdMarshaller, 4169
- activemq::wireformat::openwire::marshal::v6, 126
- activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller, 211
  - ~ActiveMQBlobMessageMarshaller, 212
  - ActiveMQBlobMessageMarshaller, 212
  - createObject, 212
  - getDataSetType, 212
  - looseMarshal, 212
  - looseUnmarshal, 213
  - tightMarshal1, 213
  - tightMarshal2, 214
  - tightUnmarshal, 214
- activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller, 252

- ~ActiveMQBytesMessageMarshaller, 253
- ActiveMQBytesMessageMarshaller, 253
- createObject, 253
- getDataStructureType, 253
- looseMarshal, 253
- looseUnmarshal, 254
- tightMarshal1, 254
- tightMarshal2, 255
- tightUnmarshal, 255
- activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller, 344
  - ~ActiveMQDestinationMarshaller, 345
  - ActiveMQDestinationMarshaller, 345
  - looseMarshal, 345
  - looseUnmarshal, 345
  - tightMarshal1, 346
  - tightMarshal2, 346
  - tightUnmarshal, 347
- activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller, 386
  - ~ActiveMQMapMessageMarshaller, 387
  - ActiveMQMapMessageMarshaller, 387
  - createObject, 387
  - getDataStructureType, 387
  - looseMarshal, 387
  - looseUnmarshal, 388
  - tightMarshal1, 388
  - tightMarshal2, 389
  - tightUnmarshal, 389
- activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller, 414
  - ~ActiveMQMessageMarshaller, 415
  - ActiveMQMessageMarshaller, 415
  - createObject, 415
  - getDataStructureType, 415
  - looseMarshal, 416
  - looseUnmarshal, 416
  - tightMarshal1, 416
  - tightMarshal2, 417
  - tightUnmarshal, 417
- activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller, 462
  - ~ActiveMQObjectMessageMarshaller, 463
  - ActiveMQObjectMessageMarshaller, 463
  - createObject, 463
  - getDataStructureType, 463
  - looseMarshal, 463
  - looseUnmarshal, 464
  - tightMarshal1, 464
  - tightMarshal2, 465
  - tightUnmarshal, 465
- activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller, 508
  - ~ActiveMQQueueMarshaller, 509
  - ActiveMQQueueMarshaller, 509
  - createObject, 509
  - getDataStructureType, 509
  - looseMarshal, 510
  - looseUnmarshal, 510
  - tightMarshal1, 510
  - tightMarshal2, 511
  - tightUnmarshal, 511
- activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller, 574
  - ~ActiveMQStreamMessageMarshaller, 575
  - ActiveMQStreamMessageMarshaller, 575
  - createObject, 575
  - getDataStructureType, 576
  - looseMarshal, 576
  - looseUnmarshal, 576
  - tightMarshal1, 577
  - tightMarshal2, 577
  - tightUnmarshal, 578
- activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller, 603
  - ~ActiveMQTempDestinationMarshaller, 604
  - ActiveMQTempDestinationMarshaller, 604
  - looseMarshal, 604
  - looseUnmarshal, 604
  - tightMarshal1, 605
  - tightMarshal2, 605
  - tightUnmarshal, 606
- activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller, 632
  - ~ActiveMQTempQueueMarshaller, 633
  - ActiveMQTempQueueMarshaller, 633
  - createObject, 633
  - getDataStructureType, 634
  - looseMarshal, 634
  - looseUnmarshal, 634
  - tightMarshal1, 635
  - tightMarshal2, 635



- tightUnmarshal, 636
- activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller, 662
  - ~ActiveMQTempTopicMarshaller, 663
  - ActiveMQTempTopicMarshaller, 663
  - createObject, 663
  - getDataStructureType, 664
  - looseMarshal, 664
  - looseUnmarshal, 664
  - tightMarshal1, 665
  - tightMarshal2, 665
  - tightUnmarshal, 666
- activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller, 688
  - ~ActiveMQTextMessageMarshaller, 690
  - ActiveMQTextMessageMarshaller, 690
  - createObject, 690
  - getDataStructureType, 690
  - looseMarshal, 690
  - looseUnmarshal, 691
  - tightMarshal1, 691
  - tightMarshal2, 692
  - tightUnmarshal, 692
- activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller, 718
  - ~ActiveMQTopicMarshaller, 719
  - ActiveMQTopicMarshaller, 719
  - createObject, 719
  - getDataStructureType, 720
  - looseMarshal, 720
  - looseUnmarshal, 720
  - tightMarshal1, 721
  - tightMarshal2, 721
  - tightUnmarshal, 722
- activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller, 800
  - ~BaseCommandMarshaller, 801
  - BaseCommandMarshaller, 801
  - looseMarshal, 801
  - looseUnmarshal, 802
  - tightMarshal1, 803
  - tightMarshal2, 804
  - tightUnmarshal, 805
- activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller, 894
  - ~BrokerIdMarshaller, 895
  - BrokerIdMarshaller, 895
  - createObject, 895
  - getDataStructureType, 896
  - looseMarshal, 896
  - looseUnmarshal, 896
  - tightMarshal1, 897
  - tightMarshal2, 897
  - tightUnmarshal, 898
- activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller, 927
  - ~BrokerInfoMarshaller, 929
  - BrokerInfoMarshaller, 929
  - createObject, 929
  - getDataStructureType, 929
  - looseMarshal, 929
  - looseUnmarshal, 930
  - tightMarshal1, 930
  - tightMarshal2, 931
  - tightUnmarshal, 931
- activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller, 1324
  - ~ConnectionControlMarshaller, 1325
  - ConnectionControlMarshaller, 1325
  - createObject, 1325
  - getDataStructureType, 1325
  - looseMarshal, 1325
  - looseUnmarshal, 1326
  - tightMarshal1, 1326
  - tightMarshal2, 1327
  - tightUnmarshal, 1327
- activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller, 1357
  - ~ConnectionErrorMarshaller, 1358
  - ConnectionErrorMarshaller, 1358
  - createObject, 1358
  - getDataStructureType, 1359
  - looseMarshal, 1359
  - looseUnmarshal, 1359
  - tightMarshal1, 1360
  - tightMarshal2, 1360
  - tightUnmarshal, 1361
- activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller, 1389
  - ~ConnectionIdMarshaller, 1390
  - ConnectionIdMarshaller, 1390
  - createObject, 1390
  - getDataStructureType, 1390
  - looseMarshal, 1390
  - looseUnmarshal, 1391
  - tightMarshal1, 1391
  - tightMarshal2, 1392
  - tightUnmarshal, 1392

- activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller, 1562
  - looseMarshal, 1562
  - looseUnmarshal, 1563
  - tightMarshal1, 1563
  - tightMarshal2, 1564
  - tightUnmarshal, 1564
- activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller, 1596
  - ~DataArrayResponseMarshaller, 1597
  - createObject, 1597
  - DataArrayResponseMarshaller, 1597
  - getDataStructureType, 1597
  - looseMarshal, 1598
  - looseUnmarshal, 1598
  - tightMarshal1, 1599
  - tightMarshal2, 1599
  - tightUnmarshal, 1600
- activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller, 1639
  - ~DataResponseMarshaller, 1640
  - createObject, 1640
  - DataResponseMarshaller, 1640
  - getDataStructureType, 1640
  - looseMarshal, 1641
  - looseUnmarshal, 1641
  - tightMarshal1, 1641
  - tightMarshal2, 1642
  - tightUnmarshal, 1642
- activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller, 1801
  - ~DestinationInfoMarshaller, 1803
  - createObject, 1803
  - DestinationInfoMarshaller, 1803
  - getDataStructureType, 1803
  - looseMarshal, 1803
  - looseUnmarshal, 1804
  - tightMarshal1, 1804
  - tightMarshal2, 1805
  - tightUnmarshal, 1805
- activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller, 1815
  - ~DiscoveryEventMarshaller, 1816
  - createObject, 1816
  - DiscoveryEventMarshaller, 1816
  - getDataStructureType, 1816
  - looseMarshal, 1817
  - looseUnmarshal, 1817
  - tightMarshal1, 1817
  - tightMarshal2, 1818
  - tightUnmarshal, 1818
- activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller, 1421
  - ~ConnectionInfoMarshaller, 1422
  - ConnectionInfoMarshaller, 1422
  - createObject, 1422
  - getDataStructureType, 1422
  - looseMarshal, 1423
  - looseUnmarshal, 1423
  - tightMarshal1, 1423
  - tightMarshal2, 1424
  - tightUnmarshal, 1424
- activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller, 1468
  - ~ConsumerControlMarshaller, 1469
  - ConsumerControlMarshaller, 1469
  - createObject, 1469
  - getDataStructureType, 1469
  - looseMarshal, 1470
  - looseUnmarshal, 1470
  - tightMarshal1, 1470
  - tightMarshal2, 1471
  - tightUnmarshal, 1471
- activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller, 1497
  - ~ConsumerIdMarshaller, 1498
  - ConsumerIdMarshaller, 1498
  - createObject, 1498
  - getDataStructureType, 1499
  - looseMarshal, 1499
  - looseUnmarshal, 1499
  - tightMarshal1, 1500
  - tightMarshal2, 1500
  - tightUnmarshal, 1501
- activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller, 1531
  - ~ConsumerInfoMarshaller, 1532
  - ConsumerInfoMarshaller, 1532
  - createObject, 1532
  - getDataStructureType, 1532
  - looseMarshal, 1533
  - looseUnmarshal, 1533
  - tightMarshal1, 1533
  - tightMarshal2, 1534
  - tightUnmarshal, 1534
- activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller, 1561
  - ~ControlCommandMarshaller, 1562
  - ControlCommandMarshaller, 1562
  - createObject, 1562
  - getDataStructureType, 1562

- activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller, 1898
- ~ExceptionResponseMarshaller, 1899
- createObject, 1899
- ExceptionResponseMarshaller, 1899
- getDataStructureType, 1899
- looseMarshal, 1899
- looseUnmarshal, 1900
- tightMarshal1, 1900
- tightMarshal2, 1901
- tightUnmarshal, 1901
- activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller, 2002
- ~FlushCommandMarshaller, 2003
- createObject, 2003
- FlushCommandMarshaller, 2003
- getDataStructureType, 2003
- looseMarshal, 2003
- looseUnmarshal, 2004
- tightMarshal1, 2004
- tightMarshal2, 2005
- tightUnmarshal, 2005
- activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller, 2162
- ~IntegerResponseMarshaller, 2164
- createObject, 2164
- getDataStructureType, 2164
- IntegerResponseMarshaller, 2164
- looseMarshal, 2164
- looseUnmarshal, 2165
- tightMarshal1, 2165
- tightMarshal2, 2166
- tightUnmarshal, 2166
- activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller, 2228
- ~JournalQueueAckMarshaller, 2229
- createObject, 2229
- getDataStructureType, 2229
- JournalQueueAckMarshaller, 2229
- looseMarshal, 2229
- looseUnmarshal, 2230
- tightMarshal1, 2230
- tightMarshal2, 2231
- tightUnmarshal, 2231
- activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller, 2269
- ~JournalTopicAckMarshaller, 2270
- createObject, 2270
- getDataStructureType, 2270
- JournalTopicAckMarshaller, 2270
- activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller, 2338
- ~KeepAliveInfoMarshaller, 2339
- createObject, 2339
- getDataStructureType, 2340
- KeepAliveInfoMarshaller, 2339
- looseMarshal, 2340
- looseUnmarshal, 2340
- tightMarshal1, 2341
- tightMarshal2, 2341
- tightUnmarshal, 2342
- activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller, 2374
- ~LastPartialCommandMarshaller, 2375
- createObject, 2375
- getDataStructureType, 2375
- LastPartialCommandMarshaller, 2375
- looseMarshal, 2375
- looseUnmarshal, 2376
- tightMarshal1, 2376
- tightMarshal2, 2377
- tightUnmarshal, 2377
- activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller, 2270
- looseUnmarshal, 2271
- tightMarshal1, 2271
- tightMarshal2, 2272
- tightUnmarshal, 2272
- activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller, 2292
- ~JournalTraceMarshaller, 2293
- createObject, 2293
- getDataStructureType, 2293
- JournalTraceMarshaller, 2293
- looseMarshal, 2293
- looseUnmarshal, 2294
- tightMarshal1, 2294
- tightMarshal2, 2295
- tightUnmarshal, 2295
- activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller, 2311
- ~JournalTransactionMarshaller, 2312
- createObject, 2312
- getDataStructureType, 2312
- JournalTransactionMarshaller, 2312
- looseMarshal, 2313
- looseUnmarshal, 2313
- tightMarshal1, 2314
- tightMarshal2, 2314
- tightUnmarshal, 2314

- activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller, 2425
  - ~LocalTransactionIdMarshaller, 2426
  - createObject, 2426
  - getDataStructureType, 2426
  - LocalTransactionIdMarshaller, 2426
  - looseMarshal, 2426
  - looseUnmarshal, 2427
  - tightMarshal1, 2427
  - tightMarshal2, 2428
  - tightUnmarshal, 2428
- activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller, 2771
  - ~MessageIdMarshaller, 2772
  - createObject, 2772
  - getDataStructureType, 2773
  - looseMarshal, 2773
  - looseUnmarshal, 2773
  - MessageIdMarshaller, 2772
  - tightMarshal1, 2774
  - tightMarshal2, 2774
  - tightUnmarshal, 2775
- activemq::wireformat::openwire::marshal::v6::MessageMarshaller, 2802
  - ~MessageMarshaller, 2804
  - looseMarshal, 2804
  - MessageAckMarshaller, 2804
  - MessageMarshaller, 2804
  - tightMarshal1, 2805
  - tightMarshal2, 2806
  - tightUnmarshal, 2806
- activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller, 2850
  - ~MessagePullMarshaller, 2851
  - createObject, 2851
  - getDataStructureType, 2851
  - looseMarshal, 2851
  - MessageDispatchMarshaller, 2852
  - MessagePullMarshaller, 2851
  - tightMarshal1, 2852
  - tightMarshal2, 2853
  - tightUnmarshal, 2853
- activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller, 2885
  - ~NetworkBridgeFilterMarshaller, 2886
  - createObject, 2886
  - getDataStructureType, 2886
  - looseMarshal, 2886
  - MessageDispatchNotificationMarshaller, 2887
  - NetworkBridgeFilterMarshaller, 2886
  - tightMarshal1, 2887
  - tightMarshal2, 2888
  - tightUnmarshal, 2888
- activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, 3005
  - ~PartialCommandMarshaller, 3007
  - createObject, 3007
  - getDataStructureType, 3007
  - looseMarshal, 3007
  - looseUnmarshal, 3008
  - PartialCommandMarshaller, 3007

- tightMarshal1, 3008
- tightMarshal2, 3009
- tightUnmarshal, 3009
- activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller, 3145
  - ~ProducerAckMarshaller, 3147
  - createObject, 3147
  - getDataStructureType, 3147
  - looseMarshal, 3147
  - looseUnmarshal, 3148
  - ProducerAckMarshaller, 3147
  - tightMarshal1, 3148
  - tightMarshal2, 3149
  - tightUnmarshal, 3149
- activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, 3177
  - ~ProducerIdMarshaller, 3178
  - createObject, 3178
  - getDataStructureType, 3179
  - looseMarshal, 3179
  - looseUnmarshal, 3179
  - ProducerIdMarshaller, 3178
  - tightMarshal1, 3180
  - tightMarshal2, 3180
  - tightUnmarshal, 3181
- activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller, 3211
  - ~ProducerInfoMarshaller, 3213
  - createObject, 3213
  - getDataStructureType, 3213
  - looseMarshal, 3213
  - looseUnmarshal, 3214
  - ProducerInfoMarshaller, 3213
  - tightMarshal1, 3214
  - tightMarshal2, 3215
  - tightUnmarshal, 3215
- activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller, 3292
  - ~RemoveInfoMarshaller, 3293
  - createObject, 3293
  - getDataStructureType, 3293
  - looseMarshal, 3294
  - looseUnmarshal, 3294
  - RemoveInfoMarshaller, 3293
  - tightMarshal1, 3294
  - tightMarshal2, 3295
  - tightUnmarshal, 3295
- activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller, 3331
  - ~RemoveSubscriptionInfoMarshaller, 3332
  - createObject, 3332
  - getDataStructureType, 3332
  - looseMarshal, 3332
  - looseUnmarshal, 3333
  - RemoveSubscriptionInfoMarshaller, 3332
  - tightMarshal1, 3333
  - tightMarshal2, 3334
  - tightUnmarshal, 3334
- activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller, 3364
  - ~ReplayCommandMarshaller, 3365
  - createObject, 3365
  - getDataStructureType, 3365
  - looseMarshal, 3366
  - looseUnmarshal, 3366
  - ReplayCommandMarshaller, 3365
  - tightMarshal1, 3366
  - tightMarshal2, 3367
  - tightUnmarshal, 3367
- activemq::wireformat::openwire::marshal::v6::ResponseMarshaller, 3413
  - ~ResponseMarshaller, 3415
  - createObject, 3415
  - getDataStructureType, 3415
  - looseMarshal, 3415
  - looseUnmarshal, 3416
  - ResponseMarshaller, 3415
  - tightMarshal1, 3416
  - tightMarshal2, 3417
  - tightUnmarshal, 3418
- activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller, 3489
  - ~SessionIdMarshaller, 3490
  - createObject, 3490
  - getDataStructureType, 3490
  - looseMarshal, 3490
  - looseUnmarshal, 3491
  - SessionIdMarshaller, 3490
  - tightMarshal1, 3491
  - tightMarshal2, 3492
  - tightUnmarshal, 3492
- activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 3508
  - ~SessionInfoMarshaller, 3510
  - createObject, 3510
  - getDataStructureType, 3510
  - looseMarshal, 3510
  - looseUnmarshal, 3511

- SessionInfoMarshaller, 3510
- tightMarshal1, 3511
- tightMarshal2, 3512
- tightUnmarshal, 3512
- activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller, 3576
  - ~ShutdownInfoMarshaller, 3577
  - createObject, 3577
  - getDataStructureType, 3577
  - looseMarshal, 3578
  - looseUnmarshal, 3578
  - ShutdownInfoMarshaller, 3577
  - tightMarshal1, 3578
  - tightMarshal2, 3579
  - tightUnmarshal, 3579
- activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller, 3803
  - ~SubscriptionInfoMarshaller, 3804
  - createObject, 3804
  - getDataStructureType, 3804
  - looseMarshal, 3804
  - looseUnmarshal, 3805
  - SubscriptionInfoMarshaller, 3804
  - tightMarshal1, 3805
  - tightMarshal2, 3806
  - tightUnmarshal, 3806
- activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller, 3955
  - ~TransactionIdMarshaller, 3957
  - looseMarshal, 3957
  - looseUnmarshal, 3957
  - tightMarshal1, 3958
  - tightMarshal2, 3958
  - tightUnmarshal, 3959
  - TransactionIdMarshaller, 3957
- activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller, 3977
  - ~TransactionInfoMarshaller, 3978
  - createObject, 3978
  - getDataStructureType, 3978
  - looseMarshal, 3978
  - looseUnmarshal, 3979
  - tightMarshal1, 3979
  - tightMarshal2, 3980
  - tightUnmarshal, 3980
  - TransactionInfoMarshaller, 3978
- activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller, 4109
  - ~WireFormatInfoMarshaller, 4110
  - createObject, 4110
  - getDataStructureType, 4110
  - looseMarshal, 4110
  - looseUnmarshal, 4111
  - tightMarshal1, 4111
  - tightUnmarshal, 4112
  - WireFormatInfoMarshaller, 4110
- activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller, 4147
  - ~XATransactionIdMarshaller, 4148
  - createObject, 4148
  - getDataStructureType, 4148
  - looseMarshal, 4149
  - looseUnmarshal, 4149
  - tightMarshal1, 4150
  - tightUnmarshal, 4150
  - XATransactionIdMarshaller, 4148
- activemq::wireformat::openwire::OpenWireFormat, 2971
  - ~OpenWireFormat, 2974
  - addMarshaller, 2975
  - createNegotiator, 2975
  - DEFAULT\_VERSION, 2984
  - destroyMarshallers, 2975
  - doUnmarshal, 2975
  - getCacheEnabled, 2977
  - getMaxInactivityDuration, 2976
  - getMaxInactivityDurationInitialDelay, 2976
  - getPreferredWireFormatInfo, 2976
  - getVersion, 2976
  - hasNegotiator, 2977
  - inReceive, 2977
  - isCacheEnabled, 2977
  - isSizePrefInfoMarshaller, 2977
  - isStackTraceEnabled, 2978
  - isTcpNoDelayEnabled, 2978
  - isTightEncodingEnabled, 2978
  - looseMarshalNestedObject, 2978
  - looseUnmarshalNestedObject, 2979
  - marshal, 2979
  - NULL\_TYPE, 2984
  - OpenWireFormat, 2974
  - renegotiateWireFormat, 2979
  - setCacheEnabled, 2980
  - setMaxInactivityDuration, 2980
  - setMaxInactivityDurationInitialDelay, 2980

- setPreferedWireFormatInfo, 2981
- setSizePrefixDisabled, 2981
- setStackTraceEnabled, 2981
- setTcpNoDelayEnabled, 2981
- setTightEncodingEnabled, 2982
- setVersion, 2982
- tightMarshalNestedObject1, 2982
- tightMarshalNestedObject2, 2982
- tightUnmarshalNestedObject, 2983
- unmarshal, 2983
- activemq::wireformat::openwire::OpenWireFormatFactory, 2984
  - ~OpenWireFormatFactory, 2985
  - createWireFormat, 2985
  - OpenWireFormatFactory, 2985
- activemq::wireformat::openwire::OpenWireFormatNegotiator, 2985
  - ~OpenWireFormatNegotiator, 2987
  - close, 2987
  - onCommand, 2987
  - oneway, 2987
  - onTransportException, 2988
  - OpenWireFormatNegotiator, 2986
  - request, 2988
  - start, 2989
- activemq::wireformat::openwire::OpenWireResponseBuilder, 2989
  - ~OpenWireResponseBuilder, 2990
  - buildIncomingCommands, 2990
  - buildResponse, 2990
  - OpenWireResponseBuilder, 2990
- activemq::wireformat::openwire::utils, 130
- activemq::wireformat::openwire::utils::BooleanStream, 863
  - ~BooleanStream, 864
  - BooleanStream, 864
  - clear, 864
  - marshal, 864, 865
  - marshalledSize, 865
  - readBoolean, 865
  - unmarshal, 865
  - writeBoolean, 865
- activemq::wireformat::openwire::utils::HexTable, 2048
  - ~HexTable, 2048
  - HexTable, 2048
  - operator[], 2048, 2049
  - size, 2049
- activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2817
  - ~MessagePropertyInterceptor, 2819
  - getBooleanProperty, 2820
  - getByteProperty, 2820
  - getDoubleProperty, 2820
  - getFloatProperty, 2820
  - getIntProperty, 2821
  - getLongProperty, 2821
  - getShortProperty, 2821
  - getStringProperty, 2821
  - MessagePropertyInterceptor, 2819
  - setBooleanProperty, 2822
  - setByteProperty, 2822
  - setDoubleProperty, 2822
  - setFloatProperty, 2822
  - setIntProperty, 2823
  - setLongProperty, 2823
  - setShortProperty, 2823
  - setStringProperty, 2823
- activemq::wireformat::stomp, 130
- activemq::wireformat::stomp::StompCommandConstants, 3738
  - ABORT, 3740
  - ACK, 3740
  - ACK\_AUTO, 3740
  - ACK\_CLIENT, 3740
  - ACK\_INDIVIDUAL, 3740
  - BEGIN, 3740
  - BYTES, 3740
  - COMMIT, 3740
  - CONNECT, 3740
  - CONNECTED, 3740
  - DISCONNECT, 3740
  - ERROR\_CMD, 3740
  - HEADER\_ACK, 3740
  - HEADER\_CLIENT\_ID, 3740
  - HEADER\_CONSUMERPRIORITY, 3740
  - HEADER\_CONTENTLENGTH, 3740
  - HEADER\_CORRELATIONID, 3740
  - HEADER\_DESTINATION, 3740
  - HEADER\_DISPATCH\_ASYNC, 3740
  - HEADER\_EXCLUSIVE, 3740
  - HEADER\_EXPIRES, 3740
  - HEADER\_ID, 3740
  - HEADER\_JMSPRIORITY, 3740
  - HEADER\_LOGIN, 3740
  - HEADER\_MAXPENDINGMSGSLIMIT, 3740
  - HEADER\_MESSAGE, 3740
  - HEADER\_MESSAGEID, 3740

- HEADER\_NOLOCAL, 3740
- HEADER\_OLDSUBSCRIPTIONNAME, 3740
- HEADER\_PASSWORD, 3740
- HEADER\_PERSISTENT, 3740
- HEADER\_PREFETCHSIZE, 3740
- HEADER\_RECEIPT\_REQUIRED, 3740
- HEADER\_RECEIPTID, 3740
- HEADER\_REDELIVERED, 3740
- HEADER\_REDELIVERYCOUNT, 3740
- HEADER\_REPLYTO, 3740
- HEADER\_REQUESTID, 3740
- HEADER\_RESPONSEID, 3740
- HEADER\_RETROACTIVE, 3740
- HEADER\_SELECTOR, 3740
- HEADER\_SESSIONID, 3740
- HEADER\_SUBSCRIPTION, 3740
- HEADER\_SUBSCRIPTIONNAME, 3740
- HEADER\_TIMESTAMP, 3740
- HEADER\_TRANSACTIONID, 3740
- HEADER\_TRANSFORMATION, 3740
- HEADER\_TRANSFORMATION\_ERROR, 3740
- HEADER\_TYPE, 3740
- MESSAGE, 3740
- QUEUE\_PREFIX, 3740
- RECEIPT, 3740
- SEND, 3740
- SUBSCRIBE, 3740
- TEMPQUEUE\_PREFIX, 3740
- TEMPTOPIC\_PREFIX, 3740
- TEXT, 3740
- TOPIC\_PREFIX, 3740
- UNSUBSCRIBE, 3740
- activemq::wireformat::stomp::StompFrame, 3741
  - ~StompFrame, 3742
  - clone, 3742
  - copy, 3743
  - fromStream, 3743
  - getBody, 3743
  - getBodyLength, 3743
  - getCommand, 3744
  - getProperties, 3744
  - getProperty, 3744
  - hasProperty, 3744
  - removeProperty, 3744
  - setBody, 3745
  - setCommand, 3745
  - setProperty, 3745
  - StompFrame, 3742
  - toStream, 3745
- activemq::wireformat::stomp::StompHelper, 3746
  - ~StompHelper, 3747
  - convertConsumerId, 3747, 3748
  - convertDestination, 3748
  - convertMessageId, 3748, 3749
  - convertProducerId, 3749
  - convertProperties, 3749, 3750
  - convertTransactionId, 3750
  - StompHelper, 3747
- activemq::wireformat::stomp::StompWireFormat, 3751
  - ~StompWireFormat, 3752
  - createNegotiator, 3752
  - getVersion, 3752
  - hasNegotiator, 3752
  - inReceive, 3753
  - marshal, 3753
  - setVersion, 3753
  - StompWireFormat, 3752
  - unmarshal, 3754
- activemq::wireformat::stomp::StompWireFormatFactory, 3754
  - ~StompWireFormatFactory, 3755
  - createWireFormat, 3755
  - StompWireFormatFactory, 3755
- activemq::wireformat::WireFormat, 4088
  - ~WireFormat, 4089
  - createNegotiator, 4089
  - getVersion, 4090
  - hasNegotiator, 4090
  - inReceive, 4090
  - marshal, 4090
  - setVersion, 4091
  - unmarshal, 4091
- activemq::wireformat::WireFormatFactory, 4092
  - ~WireFormatFactory, 4093
  - createWireFormat, 4093
- activemq::wireformat::WireFormatNegotiator, 4129
  - ~WireFormatNegotiator, 4129
  - WireFormatNegotiator, 4129
- activemq::wireformat::WireFormatRegistry, 4129
  - ~WireFormatRegistry, 4131
  - findFactory, 4131



---

getInstance, 4131	ActiveMQDestination
getWireFormatNames, 4131	activemq::commands::ActiveMQDestination,
registerFactory, 4131	315
unregisterFactory, 4132	ActiveMQDestinationMarshaller
ActiveMQBlobMessage	activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarsh
activemq::commands::ActiveMQBlobMessage, 329	
186	activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarsh
ActiveMQBlobMessageMarshaller	341
activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller,	activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarsh
195	325
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller,	activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarsh
203	333
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller,	activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarsh
191	337
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller,	activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarsh
199	345
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller,	
208	activemq::exceptions::ActiveMQException,
activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller,	348
212	ActiveMQMapMessage
ActiveMQBytesMessage	activemq::commands::ActiveMQMapMessage,
activemq::commands::ActiveMQBytesMessage, 353	
219	ActiveMQMapMessageMarshaller
ActiveMQBytesMessageMarshaller	activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMar
activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller,	370
240	activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMar
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller,	382
257	activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMar
activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller,	386
236	activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMar
activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller,	374
244	activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMar
activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller,	378
249	activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMar
activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller,	387
253	ActiveMQMessage
ActiveMQConnection	activemq::commands::ActiveMQMessage,
activemq::core::ActiveMQConnection,	391
266	ActiveMQMessageMarshaller
ActiveMQConnectionFactory	activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshall
activemq::core::ActiveMQConnectionFactory, 284	activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshall
ActiveMQConnectionMetaData	411
activemq::core::ActiveMQConnectionMetaData,	activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshall
294	394
ActiveMQConsumer	activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshall
activemq::core::ActiveMQConsumer,	402
303	activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshall
ActiveMQCPP	407
activemq::library::ActiveMQCPP, 311	activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshall

---

415  
 ActiveMQMessageTemplate  
   activemq::commands::ActiveMQMessageTemplate, 539  
   activemq::commands::ActiveMQStreamMessage, 539  
   activemq::commands::ActiveMQStreamMessageMarshaller, 539  
   activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessage, 558  
 ActiveMQObjectMessage  
   activemq::commands::ActiveMQObjectMessage, 571  
   activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessage, 571  
 ActiveMQObjectMessageMarshaller  
   activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessage, 571  
   activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessage, 571  
   activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessage, 571  
   activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessage, 571  
   activemq::wireformat::openwire::marshal::v7::ActiveMQStreamMessage, 571  
   ActiveMQTempDestination  
   activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestination, 580  
   activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestination, 580  
   activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestination, 580  
   activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestination, 580  
   activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestination, 580  
   activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestination, 580  
   activemq::wireformat::openwire::marshal::v7::ActiveMQTempDestination, 580  
 ActiveMQProducer  
   activemq::core::ActiveMQProducer, 468  
 ActiveMQProperties  
   activemq::util::ActiveMQProperties, 476  
 ActiveMQQueue  
   activemq::commands::ActiveMQQueue, 592  
   activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueue, 604  
   activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueue, 604  
   activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueue, 604  
   activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueue, 604  
   activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueue, 604  
   activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueue, 604  
   activemq::wireformat::openwire::marshal::v7::ActiveMQTempQueue, 604  
 ActiveMQQueueBrowser  
   activemq::core::ActiveMQQueueBrowser, 484  
 ActiveMQQueueMarshaller  
   activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueue, 616  
   activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueue, 616  
   activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueue, 616  
   activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueue, 616  
   activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueue, 616  
   activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueue, 616  
   activemq::wireformat::openwire::marshal::v7::ActiveMQTempQueue, 616  
 ActiveMQSession  
   activemq::core::ActiveMQSession, 517  
 ActiveMQSessionExecutor  
   activemq::core::ActiveMQSession, 532  
   activemq::core::ActiveMQSessionExecutor, 533  
 ActiveMQStreamMessage  
   activemq::commands::ActiveMQStreamMessage, 539  
   activemq::commands::ActiveMQStreamMessageMarshaller, 539  
   activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessage, 558  
   activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessage, 571  
   activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessage, 571  
   activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessage, 571  
   activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessage, 571  
   activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessage, 571  
   activemq::wireformat::openwire::marshal::v7::ActiveMQStreamMessage, 571  
   ActiveMQTempDestination  
   activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestination, 580  
   activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestination, 580  
   activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestination, 580  
   activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestination, 580  
   activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestination, 580  
   activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestination, 580  
   activemq::wireformat::openwire::marshal::v7::ActiveMQTempDestination, 580  
   ActiveMQTempQueue  
   activemq::commands::ActiveMQTempQueue, 604  
   activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueue, 604  
   activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueue, 604  
   activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueue, 604  
   activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueue, 604  
   activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueue, 604  
   activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueue, 604  
   activemq::wireformat::openwire::marshal::v7::ActiveMQTempQueue, 604  
   ActiveMQTempQueueMarshaller  
   activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueue, 616  
   activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueue, 616  
   activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueue, 616  
   activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueue, 616  
   activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueue, 616  
   activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueue, 616  
   activemq::wireformat::openwire::marshal::v7::ActiveMQTempQueue, 616  
   ActiveMQTempTopic  
   activemq::commands::ActiveMQTempTopic, 638

- ActiveMQTempTopicMarshaller 1182
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller 1933
  - activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller 62, 659
    - decaf::util::AbstractQueue, 177
  - activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller 642
    - decaf::util::List, 2410
  - activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller 646
    - decaf::util::PriorityQueue, 3120
  - activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller 655
    - decaf::util::StlSet, 3734
  - activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller 663
    - decaf::util::AbstractCollection, 163
    - decaf::util::AbstractQueue, 177
- ActiveMQTextMessage
  - activemq::commands::ActiveMQTextMessage 668
    - decaf::util::Collection, 1219
    - decaf::util::List, 2411
    - decaf::util::StlList, 3701
- ActiveMQTextMessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller 681
    - decaf::util::concurrent::atomic::AtomicInteger, 756
  - activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller 694
    - addAsResource
  - activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller 673
    - addCommand
  - activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller 677
    - addConnection
  - activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller 685
    - addCommand
  - activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller 690
    - activemq::core::ActiveMQSession, 517
    - activemq::state::SessionState, 3537
- ActiveMQTopic
  - activemq::commands::ActiveMQTopic, addDestination 698
    - activemq::cmsutil::ResourceLifecycleManager, 3377
- ActiveMQTopicMarshaller 3377
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller 711
    - activemq::core::ActiveMQConnection, 276
  - activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller 724
    - addHandler
  - activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller 702
    - additionalPredicate
  - activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller 707
    - addConsumerInfo, 1509
  - activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller 715
    - decaf::util::logging::LogManager, 2483
  - activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller 719
    - activemq::wireformat::openwire::OpenWireFormat,
- ActiveMQTransactionContext 2975
  - activemq::core::ActiveMQTransactionContext, addMessageConsumer 728
    - activemq::cmsutil::ResourceLifecycleManager, 3378
- add
  - activemq::transport::failover::CloseTransportMessageProducer

activemq::cmsutil::ResourceLifecycleManager, 3378  
 addNetworkResource Adler32, 4175  
 decaf::internal::net::Network, 2876 decaf::util::zip::Adler32, 731  
 addProducer advisory  
 activemq::core::ActiveMQConnection, 266 activemq::commands::ActiveMQDestination, 322  
 activemq::core::ActiveMQSession, 517  
 activemq::state::SessionState, 3537 ADVISORY\_PREFIX  
 activemq::commands::ActiveMQDestination, 322  
 addProducerState  
 activemq::state::TransactionState, 3991  
 addPropertyChangeListener after  
 decaf::util::logging::LogManager, 2484 decaf::util::Date, 1721  
 afterCommit  
 addResource activemq::core::Synchronization, 3827  
 decaf::internal::util::ResourceLifecycleManager, 3376  
 activemq::commands::BaseDataStructure, 838  
 address  
 decaf::net::SocketImpl, 3644 activemq::wireformat::MarshalAware, 2565  
 addressBytes  
 decaf::net::InetAddress, 2085 afterMessageIsConsumed  
 addSession activemq::core::ActiveMQConsumer, 304  
 activemq::cmsutil::ResourceLifecycleManager, 3378  
 afterRollback  
 activemq::state::ConnectionState, 1431 activemq::core::Synchronization, 3827  
 addSynchronization afterUnmarshal  
 activemq::core::ActiveMQTransactionContext, 728  
 activemq::commands::BaseDataStructure, 838  
 addTask activemq::commands::Message, 2601  
 activemq::threads::CompositeTaskRunner, 1257 activemq::commands::WireFormatInfo, 4096  
 addTempDestination activemq::wireformat::MarshalAware, 2565  
 activemq::state::ConnectionState, 1431  
 addTransactionState ALL  
 activemq::state::ConnectionState, 1431 decaf::util::logging::Level, 2407  
 addTransportListener allocate  
 activemq::core::ActiveMQConnection, 267 decaf::nio::ByteBuffer, 1056  
 decaf::nio::CharBuffer, 1152  
 decaf::nio::DoubleBuffer, 1868  
 decaf::nio::FloatBuffer, 1988  
 decaf::nio::IntBuffer, 2133  
 decaf::nio::LongBuffer, 2525  
 decaf::nio::ShortBuffer, 3563  
 activemq::transport::CompositeTransport, 1259  
 activemq::transport::failover::FailoverTransport, 1933  
 activemq::transport::failover::URIPool, 4055  
 AMQ\_CATCH\_ALL\_THROW\_CMSEXCEPTION  
 CMSEExceptionSupport.h, 4273  
 addURIs AMQ\_CATCH\_EXCEPTION\_CONVERT  
 activemq::transport::failover::URIPool, 4055 activemq/exceptions/ExceptionDefines.h, 4238  
 adjustMinimum AMQ\_CATCH\_NOTHROW  
 decaf::internal::util::TimerTaskHeap, 3918 activemq/exceptions/ExceptionDefines.h, 4239

- AMQ\_CATCH\_RETHROW
  - activemq/exceptions/ExceptionDefines.h, 4239
- AMQ\_CATCHALL\_NOTHROW
  - activemq/exceptions/ExceptionDefines.h, 4239
- AMQ\_CATCHALL\_THROW
  - activemq/exceptions/ExceptionDefines.h, 4240
- AMQCPP\_API
  - activemq/util/Config.h, 4275
- ANY\_CHILD
  - activemq::commands::ActiveMQDestination::DestinationFilter, 1779
- ANY\_DESCENDENT
  - activemq::commands::ActiveMQDestination::DestinationFilter, 1779
- anyBytes
  - decaf::net::InetAddress, 2085
- append
  - decaf::io::Writer, 4135, 4136
  - decaf::lang::Appendable, 734
  - decaf::nio::CharBuffer, 1152–1154
- AprPool
  - decaf::internal::AprPool, 736
- array
  - decaf::internal::nio::ByteBuffer, 1020
  - decaf::internal::nio::CharArrayBuffer, 1141
  - decaf::internal::nio::DoubleArrayBuffer, 1859
  - decaf::internal::nio::FloatArrayBuffer, 1980
  - decaf::internal::nio::IntArrayBuffer, 2125
  - decaf::internal::nio::LongArrayBuffer, 2517
  - decaf::internal::nio::ShortArrayBuffer, 3554
  - decaf::nio::ByteBuffer, 1056
  - decaf::nio::CharBuffer, 1154
  - decaf::nio::DoubleBuffer, 1868
  - decaf::nio::FloatBuffer, 1988
  - decaf::nio::IntBuffer, 2133
  - decaf::nio::LongBuffer, 2525
  - decaf::nio::ShortBuffer, 3563
- arraycopy
  - decaf::lang::System, 3840, 3841
- arrayOffset
  - decaf::internal::nio::ByteBuffer, 1020
  - decaf::internal::nio::CharArrayBuffer, 1142
  - decaf::internal::nio::DoubleArrayBuffer, 1860
  - decaf::internal::nio::FloatArrayBuffer, 1980
  - decaf::internal::nio::IntArrayBuffer, 2125
  - decaf::internal::nio::LongArrayBuffer, 2517
  - decaf::internal::nio::ShortArrayBuffer, 3554
  - decaf::nio::ByteBuffer, 1057
  - decaf::nio::CharBuffer, 1155
  - decaf::nio::DoubleBuffer, 1869
  - decaf::nio::FloatBuffer, 1989
  - decaf::nio::IntBuffer, 2134
  - decaf::nio::LongBuffer, 2526
  - decaf::nio::ShortBuffer, 3564
- ArrayPointer
  - decaf::lang::ArrayPointer, 739, 740
- arrival
  - activemq::commands::Message, 2613
- asCharBuffer
  - decaf::internal::nio::ByteBuffer, 1021
  - decaf::nio::ByteBuffer, 1057
- asciiToModifiedUtf8
  - activemq::util::MarshallingSupport, 2572
- asDoubleBuffer
  - decaf::internal::nio::ByteBuffer, 1021
  - decaf::nio::ByteBuffer, 1057
- asFloatBuffer
  - decaf::internal::nio::ByteBuffer, 1021
  - decaf::nio::ByteBuffer, 1058
- asIntBuffer
  - decaf::internal::nio::ByteBuffer, 1022
  - decaf::nio::ByteBuffer, 1058
- asLongBuffer
  - decaf::internal::nio::ByteBuffer, 1022
  - decaf::nio::ByteBuffer, 1059
- asReadOnlyBuffer
  - decaf::internal::nio::ByteBuffer, 1022

- decaf::internal::nio::CharArrayBuffer, available
  - 1142
- decaf::internal::nio::DoubleArrayBuffer,
  - 1860
- decaf::internal::nio::FloatArrayBuffer,
  - 1981
- decaf::internal::nio::IntArrayBuffer, 2126
- decaf::internal::nio::LongArrayBuffer,
  - 2518
- decaf::internal::nio::ShortArrayBuffer,
  - 3555
- decaf::nio::ByteBuffer, 1059
- decaf::nio::CharBuffer, 1155
- decaf::nio::DoubleBuffer, 1869
- decaf::nio::FloatBuffer, 1989
- decaf::nio::IntBuffer, 2134
- decaf::nio::LongBuffer, 2526
- decaf::nio::ShortBuffer, 3564
- Assert
  - zutil.h, 4639
- asShortBuffer
  - decaf::internal::nio::ByteBuffer, availableProcessors
    - 1023
  - decaf::nio::ByteBuffer, 1059
- AsyncSignalReadErrorTask
  - activemq::transport::inactivity::InactivityMonitor, 1566, 1567
  - 2069
- AsyncWriteTask
  - activemq::transport::inactivity::InactivityMonitor, 1566, 1567
  - 2069
- atEOF
  - decaf::util::zip::InflaterInputStream, 2104
- AtomicBoolean
  - decaf::util::concurrent::atomic::AtomicBoolean, 1929
  - 746
- AtomicInteger
  - decaf::util::concurrent::atomic::AtomicInteger, 1288
  - 750
- AtomicRefCounter
  - decaf::util::concurrent::atomic::AtomicRefCounter, 1289
  - 755
- AtomicReference
  - back
    - decaf::util::concurrent::atomic::AtomicReference, 3725
  - 757
- AUTO\_ACKNOWLEDGE
  - cms::Session, 3464
- avail\_in
  - z\_stream\_s, 4175
- avail\_out
  - z\_stream\_s, 4175
- decaf::internal::io::StandardInputStream,
  - 3689
- decaf::internal::net::ssl::openssl::OpenSSLSocket,
  - 2946
- decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream,
  - 2967
- decaf::internal::net::tcp::TcpSocket, 3854
- decaf::internal::net::tcp::TcpSocketInputStream,
  - 3861
- decaf::io::BlockingByteArrayInputStream,
  - 847
- decaf::io::BufferedInputStream, 946
- decaf::io::ByteArrayInputStream, 1043
- decaf::io::FilterInputStream, 1953
- decaf::io::InputStream, 2107
- decaf::io::PushbackInputStream, 3234
- decaf::net::SocketImpl, 3638
- decaf::util::zip::InflaterInputStream, 2101
- availablePermits
  - decaf::util::concurrent::Semaphore, 3440
- availableProcessors
  - decaf::lang::System, 3842
- await
  - decaf::util::concurrent::CountDownLatch, 1566, 1567
  - decaf::util::concurrent::locks::Condition, 1285, 1286
  - decaf::util::concurrent::locks::Condition, 1286
- awaitTermination
  - decaf::util::concurrent::ExecutorService, 1286
- awaitUninterruptibly
  - decaf::util::concurrent::locks::Condition, 1288
- awaitUntil
  - decaf::util::concurrent::locks::Condition, 1289
- back
  - decaf::util::StlQueue, 3725
- inflate\_state, 2087
- BackupTransport
  - activemq::transport::failover::BackupTransport, 760
  - activemq::transport::failover::BackupTransportPool, 764
- BackupTransportPool

- activemq::transport::failover::BackupTransportPool, 763
- activemq::wireformat::stomp::StompCommandConstants, 3740
- BAD
  - inflate.h, 4627
- base\_dist
  - trees.h, 4629
- base\_length
  - trees.h, 4630
- BaseCommand
  - activemq::commands::BaseCommand, 766
- BaseCommandMarshaller
  - activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller, 787
  - activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller, 808
  - activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller, 772
  - activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller, 780
  - activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller, 794
  - activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller, 801
- before
  - decaf::util::Date, 1721
- beforeEnd
  - activemq::core::Synchronization, 3827
- beforeMarshal
  - activemq::commands::ActiveMQMapMessage, 353
  - activemq::commands::ActiveMQTextMessage, 668
  - activemq::commands::BaseDataStructure, 839
  - activemq::commands::Message, 2601
  - activemq::commands::WireFormatInfo, 4096
  - activemq::wireformat::MarshalAware, 2565
- beforeMessageIsConsumed
  - activemq::core::ActiveMQConsumer, 304
- beforeUnmarshal
  - activemq::commands::BaseDataStructure, 839
  - activemq::wireformat::MarshalAware, 2565
- BEGIN
  - begin
    - activemq::core::ActiveMQTransactionContext, 728
  - BEST\_COMPRESSION
    - decaf::util::zip::Deflater, 1768
  - BEST\_SPEED
    - decaf::util::zip::Deflater, 1768
  - bi\_buf
    - internal\_state, 2191
  - bi\_valid
    - internal\_state, 2191
  - BIG\_STRING\_TYPE
    - activemq::util::PrimitiveValueNode, 3104
  - BINARY\_MIME\_TYPE
    - activemq::commands::ActiveMQBlobMessage, 189
  - bind
    - decaf::internal::net::tcp::TcpSocket, 3854
    - decaf::net::ServerSocket, 3452
    - decaf::net::Socket, 3614
    - decaf::net::SocketImpl, 3638
  - BindException
    - decaf::net::BindException, 843, 844
  - bitCount
    - decaf::lang::Integer, 2147
    - decaf::lang::Long, 2498
  - bits
    - code, 1216
    - inflate\_state, 2087
  - BL\_CODES
    - inflate.h, 4622
  - bl\_count
    - internal\_state, 2191
  - bl\_desc
    - internal\_state, 2191
  - bl\_tree
    - internal\_state, 2191
  - block\_start
    - internal\_state, 2191
  - BLOCKED
    - decaf::lang::Thread, 3880
  - BlockingByteArrayInputStream
    - decaf::io::BlockingByteArrayInputStream, 846
  - Boolean
    - decaf::lang::Boolean, 857
  - BOOLEAN\_TYPE
    - activemq::util::PrimitiveValueNode, 3104

Generated on Tue Mar 8 2011 17:10:05 for activemq-cpp-3.2.5 by Doxygen



- BufferOverflowException
  - decaf::nio::BufferOverflowException, 965, 966
- BufferUnderflowException
  - decaf::nio::BufferUnderflowException, 967, 968
- buildIncomingCommands
  - activemq::transport::mock::ResponseBuilder, 3383
  - activemq::wireformat::openwire::OpenWireResponseBuilder, 2990
- buildMessage
  - decaf::lang::Exception, 1889
- buildResponse
  - activemq::transport::mock::ResponseBuilder, 3383
  - activemq::wireformat::openwire::OpenWireResponseBuilder, 2990
- BUSY\_STATE
  - deflate.h, 4623
- Byte
  - decaf::lang::Byte, 971
  - zconf.h, 4632
- BYTE\_ARRAY\_TYPE
  - activemq::util::PrimitiveValueNode, 3104
- BYTE\_TYPE
  - activemq::util::PrimitiveValueNode, 3104
- ByteArrayAdapter
  - decaf::internal::util::ByteArrayAdapter, 984–987
- ByteArrayBuffer
  - decaf::internal::nio::ByteArrayBuffer, 1018, 1019
- ByteArrayInputStream
  - decaf::io::ByteArrayInputStream, 1041, 1042
- ByteArrayOutputStream
  - decaf::io::ByteArrayOutputStream, 1047
- byteArrayValue
  - activemq::util::PrimitiveValueNode::PrimitiveValue, 3098
- ByteBuffer
  - decaf::nio::ByteBuffer, 1055
- Bytcf
  - zconf.h, 4632
- BYTES
  - activemq::wireformat::stomp::StompCommandConstants, 3740
- bytesToInt
  - decaf::net::InetAddress, 2080
- byteValue
  - activemq::util::PrimitiveValueNode::PrimitiveValue, 3098
  - decaf::lang::Byte, 972
  - decaf::lang::Character, 1129
  - decaf::lang::Double, 1844
  - decaf::lang::Float, 1964
  - decaf::lang::Integer, 2147
  - decaf::lang::Long, 2499
  - decaf::lang::Short, 3541
- CachedConsumer
  - activemq::cmsutil::CachedConsumer, 1099
- CachedProducer
  - activemq::util::CachedProducer, 1103
- call
  - decaf::util::concurrent::Callable, 1109
- cancel
  - decaf::util::concurrent::Future, 2030
  - decaf::util::Timer, 3904
  - decaf::util::TimerTask, 3915
- CancellationException
  - decaf::util::concurrent::CancellationException, 1110, 1111
- capacity
  - decaf::nio::Buffer, 939
- cause
  - decaf::lang::Exception, 1893
- ceil
  - decaf::lang::Math, 2579
- CertificateEncodingException
  - decaf::security::cert::CertificateEncodingException, 1117
- CertificateException
  - decaf::security::cert::CertificateException, 1119
- CertificateExpiredException
  - decaf::security::cert::CertificateExpiredException, 1121
- CertificateNotYetValidException
  - decaf::security::cert::CertificateNotYetValidException, 1123
- CertificateParsingException
  - decaf::security::cert::CertificateParsingException, 1125
- CHAR\_TYPE
  - activemq::util::PrimitiveValueNode, 3104
- Character
  - decaf::lang::Character, 1129

- decaf::lang::Character, 1128
- CharArrayBuffer
  - decaf::internal::nio::CharArrayBuffer, 1140, 1141
- charAt
  - decaf::lang::CharSequence, 1168
  - decaf::lang::String, 3777
  - decaf::nio::CharBuffer, 1155
- CharBuffer
  - decaf::nio::CharBuffer, 1152
- charf
  - zconf.h, 4632
- charValue
  - activemq::util::PrimitiveValueNode::PrimitiveValueNode, 3098
- CHECK
  - inflate.h, 4627
- check
  - inflate\_state, 2087
- checkClosed
  - decaf::io::InputStreamReader, 2118
  - decaf::io::OutputStreamWriter, 3001
  - decaf::net::ServerSocket, 3453
  - decaf::net::Socket, 3614
- checkConnectionFactory
  - activemq::cmsutil::CmsAccessor, 1185
- checkDestinationResolver
  - activemq::cmsutil::CmsDestinationAccessor, 1189
- CheckedInputStream
  - decaf::util::zip::CheckedInputStream, 1170
- CheckedOutputStream
  - decaf::util::zip::CheckedOutputStream, 1173
- checkMapIsUnmarshalled
  - activemq::commands::ActiveMQMapMessage, 353
- checkResult
  - decaf::internal::net::tcp::TcpSocket, 3855
- checkShutdown
  - activemq::state::ConnectionState, 1431
  - activemq::state::SessionState, 3537
  - activemq::state::TransactionState, 3991
- checkValidity
  - decaf::security::cert::X509Certificate, 4142
- ClassCastException
  - decaf::lang::exceptions::ClassCastException, 1177, 1178
- ClassName
  - activemq::commands::BrokerError::StackTraceElement, 3686
- cleanup
  - decaf::internal::AprPool, 736
- clear
  - activemq::core::ActiveMQSessionExecutor, 533
  - activemq::core::MessageDispatchChannel, 2685
  - activemq::util::ActiveMQProperties, 476
  - activemq::util::PrimitiveValueNode, 3107
  - activemq::wireformat::openwire::utils::BooleanStream, 1864
  - cms::CMSProperties, 1197
  - decaf::internal::util::ByteArrayAdapter, 988
  - decaf::nio::Buffer, 939
  - decaf::util::AbstractCollection, 163
  - decaf::util::AbstractQueue, 178
  - decaf::util::Collection, 1220
  - decaf::util::concurrent::ConcurrentStlMap, 1270
  - decaf::util::concurrent::SynchronousQueue, 3830
  - decaf::util::Map, 2540
  - decaf::util::PriorityQueue, 3120
  - decaf::util::Properties, 3219
  - decaf::util::StlList, 3702
  - decaf::util::StlMap, 3713
  - decaf::util::StlQueue, 3725
  - decaf::util::StlSet, 3735
- clearBody
  - activemq::commands::ActiveMQBytesMessage, 219
  - activemq::commands::ActiveMQMapMessage, 353
  - activemq::commands::ActiveMQMessageTemplate, 422
  - activemq::commands::ActiveMQStreamMessage, 539
  - activemq::commands::ActiveMQTextMessage, 668
  - cms::Message, 2619
- clearMessagesInProgress
  - activemq::core::ActiveMQConsumer, 304
  - activemq::core::ActiveMQSession, 518
  - activemq::core::ActiveMQSessionExecutor, 533

- clearProperties
  - activemq::commands::ActiveMQMessageTemplate, 422
  - cms::Message, 2620
- clearProperty
  - decaf::lang::System, 3842
- CLIENT\_ACKNOWLEDGE
  - cms::Session, 3464
- clientId
  - activemq::commands::ConnectionInfo, 1399
  - activemq::commands::JournalTopicAck, 2256
  - activemq::commands::RemoveSubscriptionInfo, 3318
  - activemq::commands::SubscriptionInfo, 3786
- clientMaster
  - activemq::commands::ConnectionInfo, 1399
- clockSequence
  - decaf::util::UUID, 4083
- clone
  - activemq::commands::ActiveMQBlobMessage, 186
  - activemq::commands::ActiveMQBytesMessage, 219
  - activemq::commands::ActiveMQMapMessage, 354
  - activemq::commands::ActiveMQMessage, 391
  - activemq::commands::ActiveMQObjectMessage, 439
  - activemq::commands::ActiveMQQueue, 480
  - activemq::commands::ActiveMQStreamMessage, 539
  - activemq::commands::ActiveMQTempQueue, 608
  - activemq::commands::ActiveMQTempTopic, 638
  - activemq::commands::ActiveMQTextMessage, 668
  - activemq::commands::ActiveMQTopic, 698
  - activemq::core::policies::DefaultPrefetchPolicy, 1727
  - activemq::core::policies::DefaultRedeliveryPolicy, 1731
  - activemq::core::PrefetchPolicy, 3064
  - activemq::core::RedeliveryPolicy, 3269
  - activemq::exceptions::ActiveMQException, 349
  - activemq::exceptions::BrokerException, 874
  - activemq::util::ActiveMQProperties, 476
  - activemq::wireformat::stomp::StompFrame, 3742
  - cms::BytesMessage, 1083
  - cms::CMSProperties, 1197
  - cms::Destination, 1778
  - cms::Message, 2620
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2928
  - decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2958
  - decaf::io::EOFException, 1883
  - decaf::io::InterruptedIOException, 2198
  - decaf::io::IOException, 2212
  - decaf::io::UnsupportedEncodingException, 4027
  - decaf::io::UTFDataFormatException, 4080
  - decaf::lang::ArrayPointer, 740
  - decaf::lang::Exception, 1889
  - decaf::lang::exceptions::ClassCastException, 1179
  - decaf::lang::exceptions::IllegalArgumentException, 2056
  - decaf::lang::exceptions::IllegalMonitorStateException, 2059
  - decaf::lang::exceptions::IllegalStateException, 2062
  - decaf::lang::exceptions::IllegalThreadStateException, 2065
  - decaf::lang::exceptions::IndexOutOfBoundsException, 2072
  - decaf::lang::exceptions::InterruptedException, 2196
  - decaf::lang::exceptions::InvalidStateException, 2210
  - decaf::lang::exceptions::NoSuchElementException, 2912
  - decaf::lang::exceptions::NullPointerException, 2923
  - decaf::lang::exceptions::NumberFormatException, 3423
  - decaf::lang::exceptions::RuntimeException, 3423

- decaf::lang::exceptions::UnsupportedOperationException, 4030
- decaf::lang::Throwable, 3896
- decaf::net::BindException, 844
- decaf::net::ConnectException, 1296
- decaf::net::HttpRetryException, 2051
- decaf::net::MalformedURLException, 2538
- decaf::net::NoRouteToHostException, 2907
- decaf::net::PortUnreachableException, 3062
- decaf::net::ProtocolException, 3230
- decaf::net::SocketException, 3629
- decaf::net::SocketTimeoutException, 3652
- decaf::net::UnknownHostException, 4022
- decaf::net::UnknownServiceException, 4024
- decaf::net::URISyntaxException, 4063
- decaf::nio::BufferOverflowException, 966
- decaf::nio::BufferUnderflowException, 969
- decaf::nio::InvalidMarkException, 2206
- decaf::nio::ReadOnlyBufferException, 3263
- decaf::security::cert::CertificateEncodingException, 1118
- decaf::security::cert::CertificateException, 1120
- decaf::security::cert::CertificateExpiredException, 1122
- decaf::security::cert::CertificateNotYetValidException, 1124
- decaf::security::cert::CertificateParsingException, 1126
- decaf::security::GeneralSecurityException, 2037
- decaf::security::InvalidKeyException, 2203
- decaf::security::KeyException, 2368
- decaf::security::KeyManagementException, 2371
- decaf::security::NoSuchAlgorithmException, 2910
- decaf::security::NoSuchProviderException, 2915
- decaf::security::SignatureException, 3604
- decaf::util::concurrent::BrokenBarrierException, 868
- decaf::util::concurrent::CancellationException, 1112
- decaf::util::concurrent::ExecutionException, 1926
- decaf::util::concurrent::RejectedExecutionException, 3282
- decaf::util::concurrent::TimeoutException, 3901
- decaf::util::Properties, 3219
- decaf::util::zip::DataFormatException, 1602
- decaf::util::zip::ZipException, 4178
- cloneDataStructure
- activemq::commands::ActiveMQBlobMessage, 186
- activemq::commands::ActiveMQBytesMessage, 219
- activemq::commands::ActiveMQDestination, 315
- activemq::commands::ActiveMQMapMessage, 354
- activemq::commands::ActiveMQMessage, 391
- activemq::commands::ActiveMQObjectMessage, 439
- activemq::commands::ActiveMQQueue, 481
- activemq::commands::ActiveMQStreamMessage, 539
- activemq::commands::ActiveMQTempDestination, 580
- activemq::commands::ActiveMQTempQueue, 608
- activemq::commands::ActiveMQTempTopic, 638
- activemq::commands::ActiveMQTextMessage, 669
- activemq::commands::ActiveMQTopic, 698
- activemq::commands::BooleanExpression, 862
- activemq::commands::BrokerError, 870
- activemq::commands::BrokerId, 876
- activemq::commands::BrokerInfo, 904
- activemq::commands::ConnectionControl, 1303
- activemq::commands::ConnectionError, 1333

- activemq::commands::ConnectionId, 1366
- activemq::commands::ConnectionInfo, 1395
- activemq::commands::ConsumerControl, 1443
- activemq::commands::ConsumerId, 1474
- activemq::commands::ConsumerInfo, 1504
- activemq::commands::ControlCommand, 1537
- activemq::commands::DataArrayResponse, 1573
- activemq::commands::DataResponse, 1633
- activemq::commands::DataStructure, 1714
- activemq::commands::DestinationInfo, 1781
- activemq::commands::DiscoveryEvent, 1813
- activemq::commands::ExceptionResponse, 1896
- activemq::commands::FlushCommand, 2000
- activemq::commands::IntegerResponseClose, 2161
- activemq::commands::JournalQueueAck, 2225
- activemq::commands::JournalTopicAck, 2253
- activemq::commands::JournalTrace, 2282
- activemq::commands::JournalTransaction, 2309
- activemq::commands::KeepAliveInfo, 2336
- activemq::commands::LastPartialCommand, 2372
- activemq::commands::LocalTransactionId, 2422
- activemq::commands::Message, 2601
- activemq::commands::MessageAck, 2644
- activemq::commands::MessageDispatch, 2680
- activemq::commands::MessageDispatchNotification, 2717
- activemq::commands::MessageId, 2752
- activemq::commands::MessagePull, 2825
- activemq::commands::NetworkBridgeFilter, 2878
- activemq::commands::PartialCommand, 3003
- activemq::commands::ProducerAck, 3126
- activemq::commands::ProducerId, 3158
- activemq::commands::ProducerInfo, 3187
- activemq::commands::RemoveInfo, 3285
- activemq::commands::RemoveSubscriptionInfo, 3315
- activemq::commands::ReplayCommand, 3345
- activemq::commands::Response, 3380
- activemq::commands::SessionId, 3478
- activemq::commands::SessionInfo, 3506
- activemq::commands::ShutdownInfo, 3574
- activemq::commands::SubscriptionInfo, 3783
- activemq::commands::TransactionId, 3933
- activemq::commands::TransactionInfo, 3961
- activemq::commands::WireFormatInfo, 4097
- activemq::commands::XATransactionId, 4144
- activemq::cmsutil::CachedConsumer, 1099
- activemq::cmsutil::CachedProducer, 1103
- activemq::cmsutil::PooledSession, 3045
- activemq::commands::ActiveMQTempDestination, 580
- activemq::commands::ConnectionControl, 1306
- activemq::commands::ConsumerControl, 1446
- activemq::core::ActiveMQConnection, 267
- activemq::core::ActiveMQConsumer, 304
- activemq::core::ActiveMQProducer, 468
- activemq::core::ActiveMQQueueBrowser, 484
- activemq::core::ActiveMQSession, 518
- activemq::core::ActiveMQSessionExecutor, 533
- activemq::core::MessageDispatchChannel, 2685
- activemq::transport::correlator::ResponseCorrelator, 3386

activemq::transport::failover::FailoverTransport, 1933  
 activemq::transport::inactivity::InactivityMonitor, 2067  
 activemq::transport::IOTransport, 2215  
 activemq::transport::mock::MockTransport, 2857  
 activemq::transport::tcp::TcpTransport, 3867  
 activemq::transport::TransportFilter, 4007  
 activemq::wireformat::openwire::OpenWireFormat, 2987  
 cms::Closeable, 1180  
 cms::Connection, 1298  
 cms::Session, 3464  
 decaf::internal::io::StandardErrorOutputStream, 3687  
 decaf::internal::io::StandardOutputStream, 3690  
 decaf::internal::net::ssl::openssl::OpenSSLSSLContext, 2946  
 decaf::internal::net::ssl::openssl::OpenSSLSSLContextImpl, 2968  
 decaf::internal::net::ssl::openssl::OpenSSLSSLContextOutput, 2970  
 decaf::internal::net::tcp::TcpSocket, 3855  
 decaf::internal::net::tcp::TcpSocketInputStream, 3862  
 decaf::internal::net::tcp::TcpSocketOutputStream, 3864  
 decaf::io::BlockingByteArrayInputStream, 847  
 decaf::io::BufferedInputStream, 946  
 decaf::io::Closeable, 1181  
 decaf::io::FilterInputStream, 1954  
 decaf::io::FilterOutputStream, 1959  
 decaf::io::InputStream, 2108  
 decaf::io::InputStreamReader, 2118  
 decaf::io::OutputStream, 2993  
 decaf::io::OutputStreamWriter, 3001  
 decaf::net::ServerSocket, 3453  
 decaf::net::Socket, 3614  
 decaf::net::SocketImpl, 3639  
 decaf::util::logging::ConsoleHandler, 1440  
 decaf::util::logging::StreamHandler, 3758  
 decaf::util::zip::DeflaterOutputStream, 1772  
 decaf::util::zip::InflaterInputStream, 2102  
 CLOSE\_FAILURE  
 decaf::util::logging::ErrorManager, 1885  
 closed  
 decaf::io::FilterInputStream, 1957  
 decaf::io::FilterOutputStream, 1961  
 CloseTransportsTask  
 activemq::transport::failover::CloseTransportsTask, 1182  
 activemq::commands::Message, 2613  
 cms, 131  
 WireFormatNegotiator, CMS\_API, 4275  
 cms::BytesMessage, 1079  
 ~BytesMessage, 1083  
 clone, 1083  
 getBodyBytes, 1083  
 getBodyLength, 1084  
 readBoolean, 1084  
 readByte, 1084  
 readBytes, 1085, 1086  
 readChar, 1087  
 readDouble, 1087  
 readFloat, 1087  
 readInt, 1088  
 readLong, 1088  
 readShort, 1089  
 readString, 1089  
 readUnsignedShort, 1090  
 readUTF, 1090  
 reset, 1091  
 setBodyBytes, 1091  
 writeBoolean, 1092  
 writeByte, 1092  
 writeBytes, 1093  
 writeChar, 1093  
 writeDouble, 1094  
 writeFloat, 1094  
 writeInt, 1095  
 writeLong, 1095  
 writeShort, 1095  
 writeString, 1096  
 writeUnsignedShort, 1096  
 writeUTF, 1097  
 cms::Closeable, 1179  
 ~Closeable, 1180  
 close, 1180  
 cms::CMSException, 1190  
 ~CMSException, 1192  
 CMSException, 1192  
 getCause, 1192

- getMessage, 1192
- getStackTrace, 1193
- getStackTraceString, 1193
- printStackTrace, 1193
- setMark, 1193
- what, 1194
- cms::CMSProperties, 1195
  - ~CMSProperties, 1197
  - clear, 1197
  - clone, 1197
  - copy, 1197
  - getProperty, 1197
  - hasProperty, 1198
  - isEmpty, 1198
  - remove, 1198
  - setProperty, 1199
  - toArray, 1199
  - toString, 1199
- cms::CMSSecurityException, 1199
  - ~CMSSecurityException, 1200
  - CMSSecurityException, 1200
- cms::Connection, 1296
  - ~Connection, 1298
  - close, 1298
  - createSession, 1298
  - getClientID, 1299
  - getExceptionListener, 1299
  - getMetaData, 1299
  - setClientID, 1300
  - setExceptionListener, 1300
- cms::ConnectionFactory, 1361
  - ~ConnectionFactory, 1362
  - createCMSConnectionFactory, 1362
  - createConnection, 1363, 1364
- cms::ConnectionMetaData, 1425
  - ~ConnectionMetaData, 1426
  - getCMSMajorVersion, 1426
  - getCMSMinorVersion, 1427
  - getCMSProviderName, 1427
  - getCMSVersion, 1427
  - getCMSXPropertyNames, 1428
  - getProviderMajorVersion, 1428
  - getProviderMinorVersion, 1428
  - getProviderVersion, 1429
- cms::DeliveryMode, 1775
  - ~DeliveryMode, 1776
  - DELIVERY\_MODE, 1776
  - NON\_PERSISTENT, 1776
  - PERSISTENT, 1776
- cms::Destination, 1776
  - ~Destination, 1778
  - clone, 1778
  - copy, 1778
  - DestinationType, 1777
  - getCMSProperties, 1778
  - getDestinationType, 1778
  - QUEUE, 1777
  - TEMPORARY\_QUEUE, 1777
  - TEMPORARY\_TOPIC, 1777
  - TOPIC, 1777
- cms::ExceptionListener, 1893
  - ~ExceptionListener, 1894
  - onException, 1894
- cms::IllegalStateException, 2059
  - ~IllegalStateException, 2060
  - IllegalStateException, 2060
- cms::InvalidClientIdException, 2199
  - ~InvalidClientIdException, 2200
  - InvalidClientIdException, 2200
- cms::InvalidDestinationException, 2200
  - ~InvalidDestinationException, 2201
  - InvalidDestinationException, 2201
- cms::InvalidSelectorException, 2206
  - ~InvalidSelectorException, 2207
  - InvalidSelectorException, 2207
- cms::MapMessage, 2551
  - ~MapMessage, 2554
  - getBoolean, 2554
  - getByte, 2554
  - getBytes, 2555
  - getChar, 2555
  - getDouble, 2555
  - getFloat, 2556
  - getInt, 2556
  - getLong, 2557
  - getMapNames, 2557
  - getShort, 2557
  - getString, 2558
  - itemExists, 2558
  - setBoolean, 2558
  - setByte, 2559
  - setBytes, 2559
  - setChar, 2560
  - setDouble, 2560
  - setFloat, 2561
  - setInt, 2561
  - setLong, 2561
  - setShort, 2562
  - setString, 2562
- cms::Message, 2614

- ~Message, 2619
- acknowledge, 2619
- clearBody, 2619
- clearProperties, 2620
- clone, 2620
- getBooleanProperty, 2620
- getByteProperty, 2621
- getCMSCorrelationID, 2622
- getCMSDeliveryMode, 2622
- getCMSDestination, 2623
- getCMSExpiration, 2623
- getCMSMessageID, 2624
- getCMSPriority, 2625
- getCMSRedelivered, 2625
- getCMSReplyTo, 2626
- getCMSTimestamp, 2626
- getCMSType, 2627
- getDoubleProperty, 2628
- getFloatProperty, 2628
- getIntProperty, 2629
- getLongProperty, 2629
- getPropertyNames, 2630
- getShortProperty, 2630
- getStringProperty, 2631
- propertyExists, 2632
- setBooleanProperty, 2632
- setByteProperty, 2633
- setCMSCorrelationID, 2633
- setCMSDeliveryMode, 2634
- setCMSDestination, 2635
- setCMSExpiration, 2635
- setCMSMessageID, 2636
- setCMSPriority, 2636
- setCMSRedelivered, 2637
- setCMSReplyTo, 2637
- setCMSTimestamp, 2638
- setCMSType, 2638
- setDoubleProperty, 2639
- setFloatProperty, 2640
- setIntProperty, 2640
- setLongProperty, 2641
- setShortProperty, 2641
- setStringProperty, 2642
- cms::MessageConsumer, 2674
  - ~MessageConsumer, 2675
  - getMessageListener, 2675
  - getMessageSelector, 2675
  - receive, 2676
  - receiveNoWait, 2677
  - setMessageListener, 2677
- cms::MessageEnumeration, 2746
  - ~MessageEnumeration, 2747
  - hasMoreMessages, 2747
  - nextMessage, 2747
- cms::MessageEOFException, 2747
  - ~MessageEOFException, 2748
  - MessageEOFException, 2748
- cms::MessageFormatException, 2749
  - ~MessageFormatException, 2750
  - MessageFormatException, 2750
- cms::MessageListener, 2779
  - ~MessageListener, 2780
  - onMessage, 2780
- cms::MessageNotReadableException, 2807
  - ~MessageNotReadableException, 2808
  - MessageNotReadableException, 2808
- cms::MessageNotWriteableException, 2808
  - ~MessageNotWriteableException, 2809
  - MessageNotWriteableException, 2809
- cms::MessageProducer, 2809
  - ~MessageProducer, 2811
  - getDeliveryMode, 2811
  - getDisableMessageID, 2811
  - getDisableMessageTimeStamp, 2812
  - getPriority, 2812
  - getTimeToLive, 2813
  - send, 2813–2815
  - setDeliveryMode, 2816
  - setDisableMessageID, 2816
  - setDisableMessageTimeStamp, 2816
  - setPriority, 2817
  - setTimeToLive, 2817
- cms::ObjectMessage, 2924
  - ~ObjectMessage, 2924
- cms::Queue, 3238
  - ~Queue, 3239
  - getQueueName, 3239
- cms::QueueBrowser, 3243
  - ~QueueBrowser, 3244
  - getEnumeration, 3244
  - getMessageSelector, 3244
  - getQueue, 3245
- cms::Session, 3460
  - ~Session, 3464
  - AcknowledgeMode, 3464
  - AUTO\_ACKNOWLEDGE, 3464
  - CLIENT\_ACKNOWLEDGE, 3464
  - close, 3464
  - commit, 3465
  - createBrowser, 3465, 3466



- createBytesMessage, 3466
- createConsumer, 3467, 3468
- createDurableConsumer, 3469
- createMapMessage, 3469
- createMessage, 3470
- createProducer, 3470
- createQueue, 3470
- createStreamMessage, 3471
- createTemporaryQueue, 3471
- createTemporaryTopic, 3471
- createTextMessage, 3472
- createTopic, 3472
- DUPS\_OK\_ACKNOWLEDGE, 3464
- getAcknowledgeMode, 3473
- INDIVIDUAL\_ACKNOWLEDGE, 3464
- isTransacted, 3473
- recover, 3473
- rollback, 3474
- SESSION\_TRANSACTED, 3464
- unsubscribe, 3474
- cms::Startable, 3691
  - ~Startable, 3692
  - start, 3692
- cms::Stoppable, 3755
  - ~Stoppable, 3756
  - stop, 3756
- cms::StreamMessage, 3760
  - ~StreamMessage, 3763
  - readBoolean, 3763
  - readByte, 3763
  - readBytes, 3764, 3765
  - readChar, 3766
  - readDouble, 3766
  - readFloat, 3767
  - readInt, 3767
  - readLong, 3768
  - readShort, 3768
  - readString, 3769
  - readUnsignedShort, 3770
  - writeBoolean, 3770
  - writeByte, 3771
  - writeBytes, 3771
  - writeChar, 3772
  - writeDouble, 3772
  - writeFloat, 3773
  - writeInt, 3773
  - writeLong, 3773
  - writeShort, 3774
  - writeString, 3774
  - writeUnsignedShort, 3775
- cms::TemporaryQueue, 3871
  - ~TemporaryQueue, 3872
  - destroy, 3872
  - getQueueName, 3872
- cms::TemporaryTopic, 3873
  - ~TemporaryTopic, 3873
  - destroy, 3873
  - getTopicName, 3874
- cms::TextMessage, 3874
  - ~TextMessage, 3875
  - getText, 3875
  - setText, 3875, 3876
- cms::Topic, 3930
  - ~Topic, 3931
  - getTopicName, 3931
- cms::UnsupportedOperationException, 4030
  - ~UnsupportedOperationException, 4031
  - UnsupportedOperationException, 4031
- CMS\_API
  - cms/Config.h, 4275
- CmsAccessor
  - activemq::cmsutil::CmsAccessor, 1185
- CmsDestinationAccessor
  - activemq::cmsutil::CmsDestinationAccessor, 1189
- CMSException
  - cms::CMSException, 1192
- CMSExceptionSupport.h
  - AMQ\_CATCH\_ALL\_THROW\_CMSEXCEPTION, 4273
- CMSSecurityException
  - cms::CMSSecurityException, 1200
- CmsTemplate
  - activemq::cmsutil::CmsTemplate, 1205
- Code
  - deflate.h, 4623
- code, 1215
  - bits, 1216
  - ct\_data\_s, 1571
  - op, 1216
  - val, 1216
- CODELENS
  - inflate.h, 4627
- CODES
  - infrees.h, 4628
- codes
  - inflate\_state, 2087
- codetype
  - infrees.h, 4628
- comm\_max

- gz\_header\_s, 2039
- command
  - activemq::commands::ControlCommand, 1539
- commandId
  - activemq::commands::PartialCommand, 3005
- COMMENT
  - inflate.h, 4627
- comment
  - gz\_header\_s, 2039
- COMMENT\_STATE
  - deflate.h, 4623
- COMMIT
  - activemq::wireformat::stomp::StompCommandConstants, 3740
- commit
  - activemq::cmsutil::PooledSession, 3045
  - activemq::core::ActiveMQConsumer, 305
  - activemq::core::ActiveMQSession, 518
  - activemq::core::ActiveMQTransactionContext, 728
  - cms::Session, 3465
- compact
  - decaf::internal::nio::ByteBuffer, 1023
  - decaf::internal::nio::CharArrayBuffer, 1143
  - decaf::internal::nio::DoubleArrayBuffer, 1861
  - decaf::internal::nio::FloatArrayBuffer, 1981
  - decaf::internal::nio::IntArrayBuffer, 2126
  - decaf::internal::nio::LongArrayBuffer, 2518
  - decaf::internal::nio::ShortArrayBuffer, 3556
  - decaf::nio::ByteBuffer, 1060
  - decaf::nio::CharBuffer, 1156
  - decaf::nio::DoubleBuffer, 1870
  - decaf::nio::FloatBuffer, 1990
  - decaf::nio::IntBuffer, 2135
  - decaf::nio::LongBuffer, 2527
  - decaf::nio::ShortBuffer, 3565
- COMPARATOR
  - activemq::commands::BrokerId, 876
  - activemq::commands::ConnectionId, 1366
  - activemq::commands::ConsumerId, 1474
  - activemq::commands::LocalTransactionId, 2421
  - activemq::commands::MessageId, 2752
  - activemq::commands::ProducerId, 3158
  - activemq::commands::SessionId, 3478
  - activemq::commands::TransactionId, 3933
  - activemq::commands::XATransactionId, 4144
- comparator
  - decaf::util::PriorityQueue, 3120
- compare
  - activemq::util::IdGenerator, 2053
  - decaf::lang::ArrayPointerComparator, 745
  - decaf::lang::Double, 1844
  - decaf::lang::Float, 1965
  - decaf::lang::PointerComparator, 3041
  - decaf::util::Comparator, 1252
  - decaf::util::comparators::Less, 2400
- compareAndSet
  - decaf::util::concurrent::atomic::AtomicBoolean, 747
  - decaf::util::concurrent::atomic::AtomicInteger, 750
  - decaf::util::concurrent::atomic::AtomicReference, 757
- compareTo
  - activemq::commands::BrokerId, 876
  - activemq::commands::ConnectionId, 1366
  - activemq::commands::ConsumerId, 1474
  - activemq::commands::LocalTransactionId, 2422
  - activemq::commands::MessageId, 2752
  - activemq::commands::ProducerId, 3159
  - activemq::commands::SessionId, 3478
  - activemq::commands::TransactionId, 3934
  - activemq::commands::XATransactionId, 4145
  - decaf::lang::Boolean, 857, 858
  - decaf::lang::Byte, 972
  - decaf::lang::Character, 1129
  - decaf::lang::Comparable, 1249
  - decaf::lang::Double, 1844, 1845
  - decaf::lang::Float, 1965
  - decaf::lang::Integer, 2147
  - decaf::lang::Long, 2499
  - decaf::lang::Short, 3541, 3542

- decaf::net::URI, 4036
- decaf::nio::ByteBuffer, 1060
- decaf::nio::CharBuffer, 1156
- decaf::nio::DoubleBuffer, 1870
- decaf::nio::FloatBuffer, 1990
- decaf::nio::IntBuffer, 2135
- decaf::nio::LongBuffer, 2527
- decaf::nio::ShortBuffer, 3565
- decaf::util::concurrent::TimeUnit, 3922
- decaf::util::Date, 1721
- decaf::util::logging::Level, 2406
- decaf::util::UUID, 4083
- COMPOSITE\_SEPARATOR
  - activemq::commands::ActiveMQDestination, 322
- CompositeData
  - activemq::util::CompositeData, 1254
- CompositeTaskRunner
  - activemq::threads::CompositeTaskRunner, 1257
- compressed
  - activemq::commands::Message, 2613
- Concurrent.h
  - synchronized, 4713
  - WAIT\_INFINITE, 4713
- ConcurrentStlMap
  - decaf::util::concurrent::ConcurrentStlMap, 1269, 1270
- condition
  - decaf::util::concurrent::ConditionHandle, 1291
- ConditionHandle
  - decaf::util::concurrent::ConditionHandle, 1291
- CONFIG
  - decaf::util::logging::Level, 2407
- config
  - decaf::util::logging::Logger, 2465
- configure
  - activemq::core::PrefetchPolicy, 3064
  - activemq::core::RedeliveryPolicy, 3269
  - activemq::wireformat::openwire::marshal::v1::MarshallerFactory, 2570
  - activemq::wireformat::openwire::marshal::v2::MarshallerFactory, 2571
  - activemq::wireformat::openwire::marshal::v3::MarshallerFactory, 2568
  - activemq::wireformat::openwire::marshal::v4::MarshallerFactory, 2569
- activemq::wireformat::openwire::marshal::v5::MarshallerFactory, 2569
- activemq::wireformat::openwire::marshal::v6::MarshallerFactory, 2567
- configureSocket
  - activemq::transport::tcp::SslTransport, 3683
  - activemq::transport::tcp::TcpTransport, 3867
- CONNECT
  - activemq::wireformat::stomp::StompCommandConstants, 3740
- connect
  - activemq::transport::tcp::TcpTransport, 3867
- decaf::internal::net::ssl::openssl::OpenSSLSocket, 2947
- decaf::internal::net::tcp::TcpSocket, 3855
- decaf::net::Socket, 3614, 3615
- decaf::net::SocketImpl, 3639
- CONNECTED
  - activemq::wireformat::stomp::StompCommandConstants, 3740
- connectedBrokers
  - activemq::commands::ConnectionControl, 1306
- ConnectException
  - decaf::net::ConnectException, 1294, 1295
- connection
  - activemq::commands::ActiveMQTempDestination, 582
  - activemq::commands::Message, 2613
- CONNECTION\_ADVISORY\_PREFIX
  - activemq::commands::ActiveMQDestination, 322
- CONNECTION\_ALWAYS\_SYNC\_SEND
  - activemq::core::ActiveMQConstants, 299
- CONNECTION\_CLOSETIMEOUT
  - activemq::core::ActiveMQConstants, 299
- CONNECTION\_DISPATCH\_ASYNC
  - activemq::core::ActiveMQConstants, 300
- CONNECTION\_PRODUCER\_WINDOW\_SIZE
  - activemq::core::ActiveMQConstants, 299
- CONNECTION\_SENDTIMEOUT

- activemq::core::ActiveMQConstants, 299
- CONNECTION\_USEASYNCSND
  - activemq::core::ActiveMQConstants, 299
- CONNECTION\_USECOMPRESSION
  - activemq::core::ActiveMQConstants, 299
- ConnectionControl
  - activemq::commands::ConnectionControl, 1303
- ConnectionControlMarshaller
  - activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller, 1316
  - activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller, 1329
  - activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller, 1308
  - activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller, 1312
  - activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, 1321
  - activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller, 1325
- ConnectionError
  - activemq::commands::ConnectionError, 1333
- ConnectionErrorMarshaller
  - activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1350
  - activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller, 1337
  - activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller, 1341
  - activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller, 1346
  - activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller, 1354
  - activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller, 1358
- connectionInterruptProcessingComplete
- ConnectionId
  - activemq::commands::ConnectionId, 1366
- connectionId
  - activemq::commands::BrokerInfo, 910
  - activemq::commands::ConnectionError, 1336
  - activemq::commands::ConnectionInfo, 1399
  - activemq::commands::ConsumerId, 1476
- activemq::commands::DestinationInfo, 1784
- activemq::commands::LocalTransactionId, 2424
- activemq::commands::ProducerId, 3161
- activemq::commands::RemoveSubscriptionInfo, 3318
- activemq::commands::SessionId, 3480
- activemq::commands::TransactionInfo, 3964
- ConnectionIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller, 1382
  - activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller, 1370
  - activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller, 1374
  - activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller, 1378
  - activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller, 1386
  - activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller, 1390
- ConnectionInfo
  - activemq::commands::ConnectionInfo, 1395
- ConnectionInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller, 1404
  - activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller, 1401
  - activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller, 1405
  - activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller, 1409
  - activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller, 1418
  - activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller, 1422
- activemq::state::ConnectionStateTracker, 1435
- ConnectionState
  - activemq::state::ConnectionState, 1431
- ConnectionStateTracker
  - activemq::state::ConnectionStateTracker, 1435
- decaf::util::logging::ConsoleHandler, 1440

const	activemq::commands::ConsumerInfo,
zconf.h, 4632	1509
ConstReferenceType	activemq::commands::MessageAck, 2648
decaf::lang::ArrayPointer, 739	activemq::commands::MessageDispatch,
CONSUMER_ADVISORY_PREFIX	2683
activemq::commands::ActiveMQDestination,	activemq::commands::MessageDispatchNotification,
322	2720
CONSUMER_DISPATCHASYNC	activemq::commands::MessagePull, 2828
activemq::core::ActiveMQConstants,	ConsumerIdMarshaller
299	activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller,
CONSUMER_EXCLUSIVE	1490
activemq::core::ActiveMQConstants,	activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller,
299	1478
CONSUMER_NOLOCAL	activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller,
activemq::core::ActiveMQConstants,	1482
299	activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller,
CONSUMER_PREFECTCHSIZE	1486
activemq::core::ActiveMQConstants,	activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller,
299	1494
CONSUMER_PRIORITY	activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller,
activemq::core::ActiveMQConstants,	1498
299	ConsumerInfo
CONSUMER_RETROACTIVE	activemq::commands::ConsumerInfo,
activemq::core::ActiveMQConstants,	1504
299	ConsumerInfoMarshaller
CONSUMER_SELECTOR	activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller,
activemq::core::ActiveMQConstants,	1524
299	activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller,
ConsumerControl	1511
activemq::commands::ConsumerControl,	activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller,
1443	1515
ConsumerControlMarshaller	activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller,
activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller,	1519
1461	activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller,
activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller,	1528
1448	activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller,
activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller,	1532
1452	ConsumerState
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller,	activemq::state::ConsumerState, 1536
1456	contains
activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller,	decaf::util::AbstractCollection, 164
1465	decaf::util::Collection, 1221
activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller,	decaf::util::ControlledShutdownQueue,
1469	3830
ConsumerId	decaf::util::StlList, 3702
activemq::commands::ConsumerId, 1474	decaf::util::StlSet, 3735
consumerId	containsAll
activemq::commands::ConsumerControl,	decaf::util::AbstractCollection, 165
1446	decaf::util::Collection, 1221

decaf::util::concurrent::SynchronousQueue, activemq::exceptions::ActiveMQException,  
 3830 349  
 containsKey convertTransactionId  
 decaf::util::concurrent::ConcurrentStlMap, activemq::wireformat::stomp::StompHelper,  
 1270 3750  
 decaf::util::Map, 2541 COPY  
 decaf::util::StlMap, 3714 gzguts.h, 4625  
 containsValue inflate.h, 4627  
 decaf::util::concurrent::ConcurrentStlMap copy  
 1271 activemq::commands::ActiveMQQueue,  
 decaf::util::Map, 2542 481  
 decaf::util::StlMap, 3714 activemq::commands::ActiveMQTempQueue,  
 608  
 content activemq::commands::ActiveMQTempTopic,  
 638  
 activemq::commands::Message, 2613  
 ControlCommand 638  
 activemq::commands::ControlCommand, activemq::commands::ActiveMQTopic,  
 1537 699  
 ControlCommandMarshaller activemq::util::ActiveMQProperties, 476  
 activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller,  
 1553 activemq::wireformat::stomp::StompFrame,  
 3743  
 activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller,  
 1541 cms::CMSProperties, 1197  
 cms::Destination, 1778  
 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller,  
 1545 decaf::util::AbstractCollection, 165  
 decaf::util::concurrent::ConcurrentStlMap,  
 activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller,  
 1549 1271  
 decaf::util::Map, 2543  
 activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller,  
 1558 decaf::util::Properties, 3219  
 decaf::util::StlList, 3703  
 activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller,  
 1562 decaf::util::StlMap, 3714  
 decaf::util::StlSet, 3736  
 convert COPY\_  
 activemq::util::PrimitiveValueConverter, inflate.h, 4627  
 3099 copyDataStructure  
 decaf::util::concurrent::TimeUnit, 3923  
 convertConsumerId activemq::commands::ActiveMQBlobMessage,  
 186  
 activemq::wireformat::stomp::StompHelper, activemq::commands::ActiveMQBytesMessage,  
 3747, 3748 219  
 convertDestination activemq::commands::ActiveMQDestination,  
 3748 315  
 activemq::wireformat::stomp::StompHelper, activemq::commands::ActiveMQMapMessage,  
 3748, 3749 354  
 activemq::wireformat::stomp::StompHelper, activemq::commands::ActiveMQMessage,  
 391  
 convertProducerId activemq::commands::ActiveMQObjectMessage,  
 3749 439  
 activemq::wireformat::stomp::StompHelper, activemq::commands::ActiveMQQueue,  
 481  
 convertProperties 3749, 3750  
 activemq::wireformat::stomp::StompHelper, activemq::commands::ActiveMQStreamMessage,  
 539  
 convertToCMSException

---

activemq::commands::ActiveMQTempDestination, 580	activemq::commands::JournalQueueAck, 2225
activemq::commands::ActiveMQTempQueue, 609	activemq::commands::JournalTopicAck, 2253
activemq::commands::ActiveMQTempTopic, 638	activemq::commands::JournalTrace, 2282
activemq::commands::ActiveMQTextMessage, 669	activemq::commands::JournalTransaction, 2309
activemq::commands::ActiveMQTopic, 699	activemq::commands::KeepAliveInfo, 2336
activemq::commands::BaseCommand, 766	activemq::commands::LastPartialCommand, 2372
activemq::commands::BaseDataStructure, 839	activemq::commands::LocalTransactionId, 2422
activemq::commands::BooleanExpression, 862	activemq::commands::Message, 2602
activemq::commands::BrokerError, 870	activemq::commands::MessageAck, 2644
activemq::commands::BrokerId, 876	activemq::commands::MessageDispatch, 2680
activemq::commands::BrokerInfo, 904	activemq::commands::MessageDispatchNotification, 2717
activemq::commands::ConnectionControl, 1303	activemq::commands::MessageId, 2752
activemq::commands::ConnectionError, 1334	activemq::commands::MessagePull, 2825
activemq::commands::ConnectionId, 1367	activemq::commands::NetworkBridgeFilter, 2878
activemq::commands::ConnectionInfo, 1395	activemq::commands::PartialCommand, 3003
activemq::commands::ConsumerControl, 1443	activemq::commands::ProducerAck, 3126
activemq::commands::ConsumerId, 1474	activemq::commands::ProducerId, 3159
activemq::commands::ConsumerInfo, 1504	activemq::commands::ProducerInfo, 3187
activemq::commands::ControlCommand, 1537	activemq::commands::RemoveInfo, 3285
activemq::commands::DataArrayResponse, 1573	activemq::commands::RemoveSubscriptionInfo, 3315
activemq::commands::DataResponse, 1633	activemq::commands::ReplayCommand, 3345
activemq::commands::DataStructure, 1715	activemq::commands::Response, 3380
activemq::commands::DestinationInfo, 1781	activemq::commands::SessionId, 3478
activemq::commands::DiscoveryEvent, 1813	activemq::commands::SessionInfo, 3506
activemq::commands::ExceptionResponse, 1896	activemq::commands::ShutdownInfo, 3574
activemq::commands::FlushCommand, 2000	activemq::commands::SubscriptionInfo, 3783
activemq::commands::IntegerResponse, 2161	activemq::commands::TransactionId, 3934
	activemq::commands::TransactionInfo, 3961
	activemq::commands::WireFormatInfo, 4097
	activemq::commands::XATransactionId, 4145
	activemq::commands::Message, 2613

---

- activemq::commands::MessagePull, 2828
- activemq::commands::Response, 3382
- countDown
  - decaf::util::concurrent::CountDownLatch, 1568
- CountDownLatch
  - decaf::util::concurrent::CountDownLatch, 1566
- CounterType
  - decaf::lang::ArrayPointer, 739
  - decaf::lang::Pointer, 3035
- countTokens
  - decaf::util::StringTokenizer, 3780
- CRC32
  - decaf::util::zip::CRC32, 1569
- crc32.h
  - crc\_table, 4620
- crc\_table
  - crc32.h, 4620
- create
  - activemq::transport::failover::FailoverTransportFactory, 1944
  - activemq::transport::mock::MockTransportFactory, 2865
  - activemq::transport::tcp::TcpTransportFactory, 3870
  - activemq::transport::TransportFactory, 4003
  - activemq::util::CMSExceptionSupport, 1195
  - decaf::internal::net::tcp::TcpSocket, 3855
  - decaf::internal::util::concurrent::ConditionImpl, 1292
  - decaf::internal::util::concurrent::MutexImpl, 2873
  - decaf::net::SocketImpl, 3639
  - decaf::net::URI, 4036
- createBrowser
  - activemq::cmsutil::PooledSession, 3045, 3046
  - activemq::core::ActiveMQSession, 518, 519
  - cms::Session, 3465, 3466
- createByteBuffer
  - decaf::internal::nio::BufferFactory, 954, 955
- createBytesMessage
  - activemq::cmsutil::PooledSession, 3046
  - activemq::core::ActiveMQSession, 519
  - cms::Session, 3466
- createCachedConsumer
  - activemq::cmsutil::PooledSession, 3047
- createCachedProducer
  - activemq::cmsutil::PooledSession, 3047
- createCharBuffer
  - decaf::internal::nio::BufferFactory, 955, 956
- createCMSConnectionFactory
  - cms::ConnectionFactory, 1362
- createComposite
  - activemq::transport::failover::FailoverTransportFactory, 1944
  - activemq::transport::mock::MockTransportFactory, 2865
  - activemq::transport::tcp::TcpTransportFactory, 3870
  - activemq::transport::TransportFactory, 4004
- createConnection
  - activemq::cmsutil::CmsAccessor, 1185
  - activemq::core::ActiveMQConnectionFactory, 285, 286
  - cms::ConnectionFactory, 1363, 1364
- createConsumer
  - activemq::cmsutil::PooledSession, 3048, 3049
  - activemq::core::ActiveMQSession, 520, 521
  - cms::Session, 3467, 3468
- createDestination
  - activemq::commands::ActiveMQDestination, 316
- createDoubleBuffer
  - decaf::internal::nio::BufferFactory, 956, 957
- createDurableConsumer
  - activemq::cmsutil::PooledSession, 3049
  - activemq::core::ActiveMQSession, 521
  - cms::Session, 3469
- createFloatBuffer
  - decaf::internal::nio::BufferFactory, 958, 959
- createIntBuffer
  - decaf::internal::nio::BufferFactory, 959, 960
- createLongBuffer
  - decaf::internal::nio::BufferFactory, 961, 962
- createMapMessage
  - activemq::cmsutil::PooledSession, 3050



- activemq::core::ActiveMQSession, 521
- cms::Session, 3469
- createMessage
  - activemq::cmsutil::MessageCreator, 2678
  - activemq::cmsutil::PooledSession, 3050
  - activemq::core::ActiveMQSession, 522
  - cms::Session, 3470
- createMessageEOFException
  - activemq::util::CMSEExceptionSupport, 1195
- createMessageFormatException
  - activemq::util::CMSEExceptionSupport, 1195
- createNegotiator
  - activemq::wireformat::openwire::OpenWireFormat, 2975
  - activemq::wireformat::stomp::StompWireFormat, 3752
  - activemq::wireformat::WireFormat, 4089
- createObject
  - activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1661
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller, 195
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBFormMessageMarshaller, 240
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBFormMessageMarshaller, 370
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBFormMessageMarshaller, 398
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBFormMessageMarshaller, 446
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBFormMessageMarshaller, 492
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBFormMessageMarshaller, 558
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBFormMessageMarshaller, 616
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBFormMessageMarshaller, 651
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBFormMessageMarshaller, 681
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBFormMessageMarshaller, 711
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBFormMessageMarshaller, 887
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBFormMessageMarshaller, 920
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBFormMessageMarshaller, 1316
- activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1350
- activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller, 1382
- activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller, 1414
- activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller, 1461
- activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller, 1490
- activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller, 1524
- activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller, 1553
- activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller, 1589
- activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1657
- activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller, 1798
- activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1832
- activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller, 1920
- activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller, 2020
- activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller, 2181
- activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller, 2249
- activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller, 2278
- activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller, 2301
- activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller, 2332
- activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller, 2361
- activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller, 2396
- activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller, 2447
- activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller, 2667
- activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2709
- activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller, 2739
- activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 2776

activemq::wireformat::openwire::marshal::v1::ActiveMQTemp  
 2847 659  
 activemq::wireformat::openwire::marshal::v1::ActiveMQText  
 2902 694  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTopic  
 3029 724  
 activemq::wireformat::openwire::marshal::v1::BrokerIdMarsh  
 3151 899  
 activemq::wireformat::openwire::marshal::v1::BrokerInfoMar  
 3182 933  
 activemq::wireformat::openwire::marshal::v1::ConnectionCon  
 3200 1329  
 activemq::wireformat::openwire::marshal::v1::ConnectionErr  
 3302 1337  
 activemq::wireformat::openwire::marshal::v1::ConnectionIdM  
 3319 1370  
 activemq::wireformat::openwire::marshal::v1::ConnectionInfo  
 3352 1401  
 activemq::wireformat::openwire::marshal::v1::ConsumerCont  
 3410 1448  
 activemq::wireformat::openwire::marshal::v1::ConsumerIdMa  
 3502 1478  
 activemq::wireformat::openwire::marshal::v1::ConsumerInfo  
 3518 1511  
 activemq::wireformat::openwire::marshal::v1::ControlComma  
 3586 1541  
 activemq::wireformat::openwire::marshal::v1::DataArrayResp  
 3792 1576  
 activemq::wireformat::openwire::marshal::v1::DataResponseM  
 3969 1644  
 activemq::wireformat::openwire::marshal::v1::DestinationInfo  
 4122 1786  
 activemq::wireformat::openwire::marshal::v1::DiscoveryEven  
 4161 1820  
 activemq::wireformat::openwire::marshal::v2::ExceptionResp  
 203 1903  
 activemq::wireformat::openwire::marshal::v2::FlushCommanc  
 257 2007  
 activemq::wireformat::openwire::marshal::v2::IntegerRespons  
 383 2168  
 activemq::wireformat::openwire::marshal::v2::JournalQueueA  
 411 2233  
 activemq::wireformat::openwire::marshal::v2::JournalTopicAc  
 459 2262  
 activemq::wireformat::openwire::marshal::v2::JournalTraceM  
 505 2285  
 activemq::wireformat::openwire::marshal::v2::JournalTransac  
 571 2316  
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfol  
 629 2344

activemq::wireformat::openwire::marshal::v2::ActiveMQPartialCommandMarshaller	2383	activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller	366
activemq::wireformat::openwire::marshal::v2::ActiveMQTravisFunctionIdMarshaller	2430	activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller	394
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller	2654	activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller	442
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueDispatcherMarshaller	2692	activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	488
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller	2726	activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller	554
activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller	2756	activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller	612
activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller	2830	activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller	642
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller	2882	activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	673
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller	3011	activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	702
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	3130	activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	879
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller	3162	activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	912
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	3196	activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	1308
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	3289	activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	1341
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	3328	activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	1374
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	3357	activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	1405
activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller	3395	activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller	1452
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller	3482	activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	1482
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	3527	activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	1515
activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller	3581	activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller	1545
activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller	3808	activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller	1580
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	3986	activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	1649
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	4114	activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	1790
activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller	4152	activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller	1824
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	191	activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	1907
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	236	activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	2011

activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v3::SubscriptionInfoMarshaller, 2172  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v3::TransactionInfoMarshaller, 2241  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v3::WireFormatInfoMarshaller, 2266  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v3::XATransactionInfoMarshaller, 2289  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v4::ActiveMQBlobMarshaller, 2320  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v4::ActiveMQByteMarshaller, 2348  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v4::ActiveMQMapMarshaller, 2379  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v4::ActiveMQMessageMarshaller, 2434  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v4::ActiveMQObjectMarshaller, 2658  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v4::ActiveMQQueueMarshaller, 2696  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v4::ActiveMQStreamMarshaller, 2730  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v4::ActiveMQTempBodyMarshaller, 2768  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v4::ActiveMQTempBodyMarshaller, 2838  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v4::ActiveMQTextMarshaller, 2894  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v4::ActiveMQTopicMarshaller, 3020  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v4::BrokerIdMarshaller, 3138  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v4::BrokerInfoMarshaller, 3170  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v4::ConnectionConfirmationMarshaller, 3208  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v4::ConnectionErrorMarshaller, 3297  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v4::ConnectionIdMarshaller, 3323  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v4::ConnectionInfoMarshaller, 3361  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v4::ConsumerControlMarshaller, 3405  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v4::ConsumerIdMarshaller, 3498  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v4::ConsumerInfoMarshaller, 3522  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, openwire::marshal::v4::ControlCommandMarshaller, 3594

activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller,	3340
1584	
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller,	3348
1653	
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller,	3390
1794	
activemq::wireformat::openwire::marshal::v4::DiscoveryInfoMarshaller,	3486
1828	
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller,	3531
1916	
activemq::wireformat::openwire::marshal::v4::FlushQueueInfoMarshaller,	3598
2016	
activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller,	3800
2177	
activemq::wireformat::openwire::marshal::v4::JmsQueueAckMarshaller,	3982
2245	
activemq::wireformat::openwire::marshal::v4::JmsQueueTopicAckMarshaller,	4118
2274	
activemq::wireformat::openwire::marshal::v4::JmsQueueTraceMarshaller,	4157
2297	
activemq::wireformat::openwire::marshal::v4::JmsQueueTransactionMarshaller,	208
2328	
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller,	249
2352	
activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller,	378
2392	
activemq::wireformat::openwire::marshal::v4::LastFrameInfoMarshaller,	407
2443	
activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller,	454
2662	
activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller,	501
2704	
activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller,	567
2734	
activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller,	625
2760	
activemq::wireformat::openwire::marshal::v4::MessagePurgeMarshaller,	655
2842	
activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller,	685
2898	
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller,	715
3025	
activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller,	891
3134	
activemq::wireformat::openwire::marshal::v4::ProducerWithMarshaller,	924
3166	
activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller,	1321
3191	
activemq::wireformat::openwire::marshal::v4::RemoveWireMarshaller,	1354
3310	
activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller,	
activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller,	
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller,	
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller,	
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller,	
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller,	
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller,	
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller,	
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller,	
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller,	
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller,	
activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller,	
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller,	
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller,	
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller,	
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller,	
activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller,	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller,	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller,	
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller,	
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller,	
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller,	
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller,	
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller,	
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller,	

activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller;marshal::v5::NetworkBridge  
 1386 2890  
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller;marshal::v5::PartialCommand  
 1418 3016  
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller;marshal::v5::ProducerAckM  
 1465 3142  
 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller;marshal::v5::ProducerIdMar  
 1494 3174  
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller;marshal::v5::ProducerInfoM  
 1528 3204  
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller;marshal::v5::RemoveInfoMa  
 1558 3306  
 activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller;marshal::v5::RemoveSubscri  
 1593 3336  
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller;marshal::v5::ReplayComm  
 1636 3369  
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller;marshal::v5::ResponseMarsh  
 1807 3400  
 activemq::wireformat::openwire::marshal::v5::DiscoveryRequestMarshaller;marshal::v5::SessionIdMarsh  
 1836 3494  
 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller;marshal::v5::SessionInfoMar  
 1912 3514  
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller;marshal::v5::ShutdownInfoM  
 2024 3590  
 activemq::wireformat::openwire::marshal::v5::ImageResponseMarshaller;marshal::v5::SubscriptionInf  
 2185 3796  
 activemq::wireformat::openwire::marshal::v5::JmsQueueInfoMarshaller;marshal::v5::TransactionInfo  
 2237 3965  
 activemq::wireformat::openwire::marshal::v5::JmsQueueTopicAckMarshaller;marshal::v5::WireFormatInf  
 2258 4106  
 activemq::wireformat::openwire::marshal::v5::JmsQueueTraceMarshaller;marshal::v5::XATransaction  
 2305 4169  
 activemq::wireformat::openwire::marshal::v5::JmsQueueTransactionMarshaller;marshal::v6::ActiveMQBlob  
 2324 212  
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller;marshal::v6::ActiveMQByte  
 2356 253  
 activemq::wireformat::openwire::marshal::v5::JmsQueueTransactionMarshaller;marshal::v6::ActiveMQMap  
 2388 387  
 activemq::wireformat::openwire::marshal::v5::JmsQueueTransactionMarshaller;marshal::v6::ActiveMQMess  
 2439 415  
 activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller;marshal::v6::ActiveMQObj  
 2671 463  
 activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller;marshal::v6::ActiveMQQue  
 2700 509  
 activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller;marshal::v6::ActiveMQStrea  
 2743 575  
 activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller;marshal::v6::ActiveMQTemp  
 2764 633  
 activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller;marshal::v6::ActiveMQTemp  
 2834 663

Generated on Tue Mar 8 2011 17:10:05 for activemq-cpp-3.2.5 by Doxygen

- createQueue
  - activemq::cmsutil::PooledSession, 3051
  - activemq::core::ActiveMQSession, 522
  - cms::Session, 3470
- createRemoveCommand
  - activemq::commands::ConnectionInfo, 1395
  - activemq::commands::ConsumerInfo, 1504
  - activemq::commands::ProducerInfo, 3187
  - activemq::commands::SessionInfo, 3506
- createServerSocket
  - decaf::internal::net::DefaultServerSocketFactory, 524
  - 1736–1738
  - decaf::internal::net::ssl::DefaultSSLServerSocketFactory, 1747, 1748
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2938, 2939
  - decaf::net::ServerSocketFactory, 3458, 3459
- createSession
  - activemq::cmsutil::CmsAccessor, 1185
  - activemq::core::ActiveMQConnection, 267, 268
  - cms::Connection, 1298
- createShortBuffer
  - decaf::internal::nio::BufferFactory, 962–964
- createSocket
  - activemq::transport::tcp::SslTransport, 3683
  - activemq::transport::tcp::TcpTransport, 3868
  - decaf::internal::net::DefaultSocketFactory, 1741–1743
  - decaf::internal::net::ssl::DefaultSSLSocketFactory, 1753–1756
  - decaf::internal::net::ssl::openssl::OpenSSLSocketFactory, 2962–2965
  - decaf::net::SocketFactory, 3631–3633
  - decaf::net::ssl::SSLSocketFactory, 3680
- createSocketImpl
  - decaf::net::SocketImplFactory, 3645
- createStreamMessage
  - activemq::cmsutil::PooledSession, 3051
  - activemq::core::ActiveMQSession, 523
  - cms::Session, 3471
- createTemporaryName
  - activemq::commands::ActiveMQDestination, 316
- createTemporaryQueue
  - activemq::cmsutil::PooledSession, 3052
  - activemq::core::ActiveMQSession, 523
  - cms::Session, 3471
- createTemporaryTopic
  - activemq::cmsutil::PooledSession, 3052
  - activemq::core::ActiveMQSession, 523
  - cms::Session, 3471
- createTextMessage
  - activemq::cmsutil::PooledSession, 3052, 3053
  - activemq::core::ActiveMQSession, 523, 524
  - cms::Session, 3472
- createWireFormat
  - activemq::transport::AbstractTransportFactory, 183
  - activemq::wireformat::openwire::OpenWireFormatFactory, 2985
  - activemq::wireformat::stomp::StompWireFormatFactory, 3755
  - activemq::wireformat::WireFormatFactory, 4093
- criticalSection
  - decaf::util::concurrent::ConditionHandle, 1291
- ct\_data
  - deflate.h, 4623
  - ct\_data\_s, 1571
- code, 1571
- dad, 1571
- at, 1571
- fc, 1571
- len, 1571
- CUNSUMER\_MAXPENDINGMSGLIMIT
  - activemq::core::ActiveMQConstants, 299
- currentThread
  - decaf::lang::Thread, 3881
- currentTimeMillis
  - decaf::lang::System, 3843
- d\_buf
  - internal\_state, 2191
- d\_code



- deflate.h, 4623
- D\_CODES
  - deflate.h, 4623
- d\_desc
  - internal\_state, 2191
- Dad
  - deflate.h, 4623
- dad
  - ct\_data\_s, 1571
- data
  - activemq::commands::DataArrayResponse, 1574
  - activemq::commands::DataResponse, 1634
  - activemq::commands::PartialCommand, 3005
- data\_type
  - z\_stream\_s, 4175
- DataArrayResponse
  - activemq::commands::DataArrayResponse, 1573
- DataArrayResponseMarshaller
  - activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller, 1589
  - activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller, 1576
  - activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller, 1580
  - activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller, 1584
  - activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller, 1593
  - activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller, 1597
- DataFormatException
  - decaf::util::zip::DataFormatException, 1601, 1602
- DataInputStream
  - decaf::io::DataInputStream, 1615
- DataOutputStream
  - decaf::io::DataOutputStream, 1629
- DataResponse
  - activemq::commands::DataResponse, 1633
- DataResponseMarshaller
  - activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1657
  - activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller, 1644
- activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller, 1649
- activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller, 1653
- activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller, 1636
- activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller, 1640
- dataStructure
  - activemq::commands::Message, 2613
- Date
  - decaf::util::Date, 1720
- DAYS
  - decaf::util::concurrent::TimeUnit, 3929
- DEBUG
  - decaf::util::logging::Level, 2407
- Debug
  - decaf::util::logging, 155
- debug
  - decaf::util::logging::Logger, 2466
  - decaf::util::logging::SimpleLogger, 3606
- decaf, 134
  - decaf/lang/exceptions/ExceptionDefines.h, 4241
  - DECAF\_CATCH\_EXCEPTION\_CONVERT, 4241
  - DECAF\_CATCH\_NOTHROW, 4241
  - DECAF\_CATCH\_RETHROW, 4241
  - DECAF\_CATCHALL\_NOTHROW, 4242
  - DECAF\_CATCHALL\_THROW, 4242
  - decaf/util/Config.h, 4275
  - DECAF\_API, 4275
  - DECAF\_UNUSED, 4275
  - HAVE\_PTHREAD\_H, 4275
  - HAVE\_UUID\_H, 4275
  - HAVE\_UUID\_UUID\_H, 4275
- decaf::internal, 135
  - decaf::internal::AprPool, 735
  - ~AprPool, 736
  - AprPool, 736
  - cleanup, 736
  - getAprPool, 736
  - getGlobalPool, 736
- decaf::internal::DecafRuntime, 1723
  - ~DecafRuntime, 1724
  - DecafRuntime, 1724
  - getGlobalPool, 1724
- decaf::internal::io::DataResponseMarshaller, 1657
- decaf::internal::io::StandardErrorOutputStream, 3686
- ~StandardErrorOutputStream, 3687

- close, 3687
- doWriteArrayBounded, 3687
- doWriteByte, 3687
- flush, 3688
- StandardOutputStream, 3687
- decaf::internal::io::StandardInputStream, 3688
  - ~StandardInputStream, 3689
- available, 3689
- doReadByte, 3689
- StandardInputStream, 3689
- decaf::internal::io::StandardOutputStream, 3689
  - ~StandardOutputStream, 3690
- close, 3690
- doWriteArrayBounded, 3691
- doWriteByte, 3691
- flush, 3691
- StandardOutputStream, 3690
- decaf::internal::net, 135
- decaf::internal::net::DefaultServerSocketFactory, 1735
  - ~DefaultServerSocketFactory, 1736
  - createServerSocket, 1736–1738
  - DefaultServerSocketFactory, 1736
- decaf::internal::net::DefaultSocketFactory, 1738
  - ~DefaultSocketFactory, 1741
  - createSocket, 1741–1743
  - DefaultSocketFactory, 1741
- decaf::internal::net::Network, 2874
  - ~Network, 2876
  - addAsResource, 2876
  - addNetworkResource, 2876
  - getNetworkRuntime, 2876
  - getRuntimeLock, 2876
  - initializeNetworking, 2877
  - Network, 2876
  - shutdownNetworking, 2877
- decaf::internal::net::SocketFileDescriptor, 3634
  - ~SocketFileDescriptor, 3635
  - getValue, 3635
  - SocketFileDescriptor, 3635
- decaf::internal::net::ssl, 136
- decaf::internal::net::ssl::DefaultSSLContext, 1743
  - ~DefaultSSLContext, 1744
  - DefaultSSLContext, 1744
  - getContext, 1744
- decaf::internal::net::ssl::DefaultSSLServerSocketFactory, 1744
  - ~DefaultSSLServerSocketFactory, 1747
  - createServerSocket, 1747, 1748
  - DefaultSSLServerSocketFactory, 1747
  - getDefaultCipherSuites, 1748
  - getSupportedCipherSuites, 1749
- decaf::internal::net::ssl::DefaultSSLSocketFactory, 1749
  - ~DefaultSSLSocketFactory, 1753
  - createSocket, 1753–1756
  - DefaultSSLSocketFactory, 1753
  - getDefaultCipherSuites, 1756
  - getSupportedCipherSuites, 1756
- decaf::internal::net::ssl::openssl, 136
- decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2924
  - ~OpenSSLContextSpi, 2926
  - OpenSSLContextSpi, 2926
  - OpenSSLContextSpi, 2927
  - OpenSSLContextSpi, 2927
  - providerGetServerSocketFactory, 2926
  - providerGetSocketFactory, 2926
  - providerInit, 2927
- decaf::internal::net::ssl::openssl::OpenSSLParameters, 2927
  - ~OpenSSLParameters, 2928
  - clone, 2928
  - getEnabledCipherSuites, 2928
  - getEnabledProtocols, 2929
  - getNeedClientAuth, 2929
  - getSupportedCipherSuites, 2929
  - getSupportedProtocols, 2929
  - getUseClientMode, 2929
  - getWantClientAuth, 2929
  - setEnabledCipherSuites, 2929
  - setEnabledProtocols, 2929
  - setNeedClientAuth, 2929
  - setUseClientMode, 2929
  - setWantClientAuth, 2929
- decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2929
  - ~OpenSSLServerSocket, 2932
  - accept, 2932
  - getEnabledCipherSuites, 2933
  - getEnabledProtocols, 2933
  - getNeedClientAuth, 2933
  - getSupportedCipherSuites, 2933
  - getSupportedProtocols, 2933
  - getWantClientAuth, 2934

- OpenSSLServerSocket, 2932
- setEnabledCipherSuites, 2934
- setEnabledProtocols, 2934
- setNeedClientAuth, 2935
- setWantClientAuth, 2935
- decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2935
- ~OpenSSLServerSocketFactory, 2938
- createServerSocket, 2938, 2939
- getDefaultCipherSuites, 2939
- getSupportedCipherSuites, 2940
- OpenSSLServerSocketFactory, 2938
- decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2940
- ~OpenSSLSocket, 2946
- available, 2946
- close, 2946
- connect, 2947
- setEnabledCipherSuites, 2947
- setEnabledProtocols, 2948
- getInputStream, 2948
- getNeedClientAuth, 2948
- getOutputStream, 2949
- getSupportedCipherSuites, 2949
- getSupportedProtocols, 2949
- getUseClientMode, 2950
- getWantClientAuth, 2950
- OpenSSLSocket, 2946
- read, 2950
- sendUrgentData, 2951
- setEnabledCipherSuites, 2951
- setEnabledProtocols, 2951
- setNeedClientAuth, 2952
- setOOBInline, 2952
- setUseClientMode, 2952
- setWantClientAuth, 2953
- shutdownInput, 2953
- shutdownOutput, 2953
- startHandshake, 2954
- write, 2954
- decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2955
- ~OpenSSLSocketException, 2958
- clone, 2958
- getErrorString, 2958
- OpenSSLSocketException, 2956, 2957
- decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2959
- ~OpenSSLSocketFactory, 2962
- createSocket, 2962–2965
- getDefaultCipherSuites, 2965
- getSupportedCipherSuites, 2965
- OpenSSLSocketFactory, 2962
- decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2966
- OpenSSLSocketInputStream, 2967
- available, 2967
- close, 2968
- doReadArrayBounded, 2968
- doReadByte, 2968
- OpenSSLSocketInputStream, 2967
- skip, 2968
- decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream, 2969
- ~OpenSSLSocketOutputStream, 2970
- close, 2970
- doWriteArrayBounded, 2970
- doWriteByte, 2971
- OpenSSLSocketOutputStream, 2970
- decaf::internal::net::tcp, 137
- decaf::internal::net::tcp::TcpSocket, 3850
- ~TcpSocket, 3854
- accept, 3854
- available, 3854
- bind, 3854
- checkResult, 3855
- close, 3855
- connect, 3855
- create, 3855
- getInputStream, 3856
- getLocalAddress, 3856
- getOption, 3856
- getOutputStream, 3857
- getSocketHandle, 3857
- isClosed, 3857
- isConnected, 3857
- listen, 3857
- read, 3858
- setOption, 3858
- shutdownInput, 3859
- shutdownOutput, 3859
- Exception, 3859
- TcpSocket, 3854
- write, 3859
- decaf::internal::net::tcp::TcpSocketInputStream, 3860
- ~TcpSocketInputStream, 3861
- available, 3861
- close, 3862
- doReadArrayBounded, 3862
- doReadByte, 3862

- skip, 3862
- TcpSocketInputStream, 3861
- decaf::internal::net::tcp::TcpSocketOutputStream
  - 3863
  - ~TcpSocketOutputStream, 3864
  - close, 3864
  - doWriteArrayBounded, 3865
  - doWriteByte, 3865
  - TcpSocketOutputStream, 3864
- decaf::internal::net::URIEncoderDecoder, 4044
  - ~URIEncoderDecoder, 4045
  - decode, 4045
  - encodeOthers, 4045
  - quoteIllegal, 4045
  - URIEncoderDecoder, 4045
  - validate, 4046
  - validateSimple, 4046
- decaf::internal::net::URIHelper, 4047
  - ~URIHelper, 4049
  - isValidDomainName, 4049
  - isValidHexChar, 4049
  - isValidHost, 4049
  - isValidIP4Word, 4050
  - isValidIP6Address, 4050
  - isValidIPv4Address, 4050
  - parseAuthority, 4051
  - parseURI, 4051
  - URIHelper, 4048, 4049
  - validateAuthority, 4051
  - validateFragment, 4052
  - validatePath, 4052
  - validateQuery, 4052
  - validateScheme, 4053
  - validateSsp, 4053
  - validateUserInfo, 4053
- decaf::internal::net::URIType, 4063
  - ~URIType, 4066
  - getAuthority, 4066
  - getFragment, 4066
  - getHost, 4066
  - getPath, 4066
  - getPort, 4066
  - getQuery, 4067
  - getScheme, 4067
  - getSchemeSpecificPart, 4067
  - getSource, 4067
  - getUserInfo, 4067
  - isAbsolute, 4068
  - isOpaque, 4068
  - isServerAuthority, 4068
  - isValid, 4068
  - setAbsolute, 4068
  - setAuthority, 4069
  - setFragment, 4069
  - setHost, 4069
  - setOpaque, 4069
  - setPath, 4069
  - setPort, 4070
  - setQuery, 4070
  - setScheme, 4070
  - setSchemeSpecificPart, 4070
  - setServerAuthority, 4070
  - setSource, 4071
  - setUserInfo, 4071
  - setValid, 4071
  - URIType, 4066
- decaf::internal::nio, 138
- decaf::internal::nio::BufferFactory, 951
  - ~BufferFactory, 954
  - createByteBuffer, 954, 955
  - createCharBuffer, 955, 956
  - createDoubleBuffer, 956, 957
  - createFloatBuffer, 958, 959
  - createIntBuffer, 959, 960
  - createLongBuffer, 961, 962
  - createShortBuffer, 962–964
- decaf::internal::nio::ByteBuffer, 1004
  - ~ByteBuffer, 1019
  - array, 1020
  - arrayOffset, 1020
  - asCharBuffer, 1021
  - asDoubleBuffer, 1021
  - asFloatBuffer, 1021
  - asIntBuffer, 1022
  - asLongBuffer, 1022
  - asReadOnlyBuffer, 1022
  - asShortBuffer, 1023
  - ByteBuffer, 1018, 1019
  - compact, 1023
  - duplicate, 1024
  - get, 1024, 1025
  - getChar, 1025
  - getDouble, 1026
  - getFloat, 1027
  - getInt, 1027, 1028
  - getLong, 1028
  - getShort, 1029
  - hasArray, 1030
  - isReadOnly, 1030

- put, 1030, 1031
- putChar, 1031, 1032
- putDouble, 1032, 1033
- putFloat, 1033, 1034
- putInt, 1034, 1035
- putLong, 1035, 1036
- putShort, 1036, 1037
- setReadOnly, 1037
- slice, 1037
- decaf::internal::nio::CharArrayBuffer, 1135
  - ~CharArrayBuffer, 1141
  - \_array, 1148
  - array, 1141
  - arrayOffset, 1142
  - asReadOnlyBuffer, 1142
  - CharArrayBuffer, 1140, 1141
  - compact, 1143
  - duplicate, 1143
  - get, 1144
  - hasArray, 1144
  - isReadOnly, 1145
  - length, 1148
  - offset, 1148
  - put, 1145, 1146
  - readOnly, 1148
  - setReadOnly, 1146
  - slice, 1146
  - subSequence, 1147
- decaf::internal::nio::DoubleArrayBuffer, 1853
  - ~DoubleArrayBuffer, 1859
  - array, 1859
  - arrayOffset, 1860
  - asReadOnlyBuffer, 1860
  - compact, 1861
  - DoubleArrayBuffer, 1858, 1859
  - duplicate, 1861
  - get, 1862
  - hasArray, 1862
  - isReadOnly, 1863
  - put, 1863
  - setReadOnly, 1864
  - slice, 1864
- decaf::internal::nio::FloatArrayBuffer, 1974
  - ~FloatArrayBuffer, 1979
  - array, 1980
  - arrayOffset, 1980
  - asReadOnlyBuffer, 1981
  - compact, 1981
  - duplicate, 1981
  - FloatArrayBuffer, 1978, 1979
  - get, 1982
  - hasArray, 1983
  - isReadOnly, 1983
  - put, 1983, 1984
  - setReadOnly, 1984
  - slice, 1984
- decaf::internal::nio::IntArrayBuffer, 2119
  - ~IntArrayBuffer, 2125
  - array, 2125
  - arrayOffset, 2125
  - asReadOnlyBuffer, 2126
  - compact, 2126
  - duplicate, 2126
  - get, 2127
  - hasArray, 2128
  - IntArrayBuffer, 2123, 2124
  - isReadOnly, 2128
  - put, 2128, 2129
  - setReadOnly, 2129
  - slice, 2129
- decaf::internal::nio::LongArrayBuffer, 2511
  - ~LongArrayBuffer, 2516
  - array, 2517
  - arrayOffset, 2517
  - asReadOnlyBuffer, 2518
  - compact, 2518
  - duplicate, 2518
  - get, 2519
  - hasArray, 2520
  - isReadOnly, 2520
  - LongArrayBuffer, 2515, 2516
  - put, 2520, 2521
  - setReadOnly, 2521
  - slice, 2521
- decaf::internal::nio::ShortArrayBuffer, 3548
  - ~ShortArrayBuffer, 3554
  - array, 3554
  - arrayOffset, 3555
  - asReadOnlyBuffer, 3555
  - compact, 3556
  - duplicate, 3556
  - get, 3557
  - hasArray, 3557
  - isReadOnly, 3558
  - put, 3558
  - setReadOnly, 3559
  - ShortArrayBuffer, 3553, 3554
  - slice, 3559
- decaf::internal::security, 138

- decaf::internal::security::SecureRandomImpl, 3429
  - ~SecureRandomImpl, 3430
  - providerGenerateSeed, 3430, 3431
  - providerNextBytes, 3431
  - providerSetSeed, 3432
  - SecureRandomImpl, 3430
- decaf::internal::util, 138
- decaf::internal::util::ByteArrayAdapter, 979
  - ~ByteArrayAdapter, 987
  - ByteArrayAdapter, 984–987
  - clear, 988
  - get, 988
  - getByteArray, 988
  - getCapacity, 988
  - getChar, 988
  - getCharArray, 989
  - getCharCapacity, 989
  - getDouble, 989
  - getDoubleArray, 990
  - getDoubleAt, 990
  - getDoubleCapacity, 990
  - getFloat, 991
  - getFloatArray, 991
  - getFloatAt, 991
  - getFloatCapacity, 992
  - getInt, 992
  - getIntArray, 992
  - getIntAt, 993
  - getIntCapacity, 993
  - getLong, 993
  - getLongArray, 994
  - getLongAt, 994
  - getLongCapacity, 994
  - getShort, 994
  - getShortArray, 995
  - getShortAt, 995
  - getShortCapacity, 996
  - operator[], 996
  - put, 996
  - putChar, 997
  - putDouble, 997
  - putDoubleAt, 998
  - putFloat, 998
  - putFloatAt, 999
  - putInt, 999
  - putIntAt, 999
  - putLong, 1000
  - putLongAt, 1000
  - putShort, 1001
  - putShortAt, 1001
  - read, 1002
  - resize, 1002
  - write, 1003
- decaf::internal::util::concurrent, 139
- decaf::internal::util::concurrent::ConditionImpl, 1291
  - create, 1292
  - destroy, 1292
  - notify, 1292
  - notifyAll, 1292
  - wait, 1293
- decaf::internal::util::concurrent::MutexImpl, 2872
  - create, 2873
  - destroy, 2873
  - lock, 2873
  - trylock, 2873
  - unlock, 2874
- decaf::internal::util::concurrent::SynchronizableImpl, 3822
  - ~SynchronizableImpl, 3823
  - lock, 3823
  - notify, 3823
  - notifyAll, 3823
  - SynchronizableImpl, 3823
  - tryLock, 3824
  - unlock, 3824
  - wait, 3824, 3825
- decaf::internal::util::concurrent::Transferer, 3991
- decaf::internal::util::concurrent::TransferQueue, 3992
  - ~TransferQueue, 3993
  - transfer, 3993
  - TransferQueue, 3993
- decaf::internal::util::concurrent::TransferStack, 3994
  - ~TransferStack, 3995
  - transfer, 3995
  - TransferStack, 3995
- decaf::internal::util::GenericResource, 2037
  - ~GenericResource, 2038
  - GenericResource, 2038
  - getManaged, 2038
  - setManaged, 2038
- decaf::internal::util::HexStringParser, 2046
  - ~HexStringParser, 2047
  - HexStringParser, 2047
  - parse, 2047

- parseDouble, 2047
- parseFloat, 2047
- decaf::internal::util::Resource, 3374
  - ~Resource, 3375
- decaf::internal::util::ResourceLifecycleManager, available, 1043
  - 3375
  - ~ResourceLifecycleManager, 3376
  - addResource, 3376
  - destroyResources, 3376
  - ResourceLifecycleManager, 3376
- decaf::internal::util::TimerTaskHeap, 3916
  - ~TimerTaskHeap, 3918
  - adjustMinimum, 3918
  - decaf::util::TimerTask, 3916
  - deleteIfCancelled, 3918
  - find, 3918
  - insert, 3918
  - isEmpty, 3918
  - peek, 3919
  - remove, 3919
  - reset, 3919
  - size, 3919
  - TimerTaskHeap, 3918
- decaf::io, 139
- decaf::io::BlockingByteArrayInputStream, 845
  - ~BlockingByteArrayInputStream, 846
  - available, 847
  - BlockingByteArrayInputStream, 846
  - close, 847
  - doReadArrayBounded, 847
  - doReadByte, 847
  - setByteArray, 847
  - skip, 848
- decaf::io::BufferedInputStream, 943
  - ~BufferedInputStream, 945
  - available, 946
  - BufferedInputStream, 945
  - close, 946
  - doReadArrayBounded, 946
  - doReadByte, 946
  - mark, 946
  - markSupported, 947
  - reset, 947
  - skip, 948
- decaf::io::BufferedOutputStream, 949
  - ~BufferedOutputStream, 950
  - BufferedOutputStream, 949, 950
  - doWriteArray, 950
  - doWriteArrayBounded, 950
- doWriteByte, 950
- flush, 950
- decaf::io::ByteArrayInputStream, 1038
  - ~ByteArrayInputStream, 1042
- ByteArrayInputStream, 1041, 1042
- doReadArrayBounded, 1043
- doReadByte, 1043
- mark, 1043
- markSupported, 1044
- reset, 1044
- setByteArray, 1045
- skip, 1046
- decaf::io::ByteArrayOutputStream, 1046
  - ~ByteArrayOutputStream, 1048
  - ByteArrayOutputStream, 1047
  - doWriteArrayBounded, 1048
  - doWriteByte, 1048
  - reset, 1048
  - size, 1048
  - toByteArray, 1049
  - toString, 1049
  - writeTo, 1049
- decaf::io::Closeable, 1180
  - ~Closeable, 1181
  - close, 1181
- decaf::io::DataInput, 1603
  - ~DataInput, 1605
  - readBoolean, 1605
  - readByte, 1605
  - readChar, 1606
  - readDouble, 1606
  - readFloat, 1606
  - readFully, 1607, 1608
  - readInt, 1608
  - readLine, 1609
  - readLong, 1609
  - readShort, 1610
  - readString, 1610
  - readUnsignedByte, 1610
  - readUnsignedShort, 1611
  - readUTF, 1611
  - skipBytes, 1612
- decaf::io::DataInputStream, 1612
  - ~DataInputStream, 1615
  - DataInputStream, 1615
  - readBoolean, 1615
  - readByte, 1615
  - readChar, 1615
  - readDouble, 1616

- readFloat, 1616
- readFully, 1617
- readInt, 1618
- readLine, 1618
- readLong, 1619
- readShort, 1619
- readString, 1620
- readUnsignedByte, 1620
- readUnsignedShort, 1620
- readUTF, 1621
- skipBytes, 1621
- decaf::io::DataOutput, 1622
  - ~DataOutput, 1624
  - writeBoolean, 1624
  - writeByte, 1624
  - writeBytes, 1624
  - writeChar, 1625
  - writeChars, 1625
  - writeDouble, 1625
  - writeFloat, 1626
  - writeInt, 1626
  - writeLong, 1626
  - writeShort, 1627
  - writeUnsignedShort, 1627
  - writeUTF, 1627
- decaf::io::DataOutputStream, 1628
  - ~DataOutputStream, 1630
  - buffer, 1631
  - DataOutputStream, 1629
  - doWriteArrayBounded, 1630
  - doWriteByte, 1630
  - size, 1630
  - writeBoolean, 1630
  - writeByte, 1631
  - writeBytes, 1631
  - writeChar, 1631
  - writeChars, 1631
  - writeDouble, 1631
  - writeFloat, 1631
  - writeInt, 1631
  - writeLong, 1631
  - writeShort, 1631
  - writeUnsignedShort, 1631
  - writeUTF, 1631
  - written, 1631
- decaf::io::EOFException, 1881
  - ~EOFException, 1883
  - clone, 1883
  - EOFException, 1882, 1883
- decaf::io::FileDescriptor, 1947
  - ~FileDescriptor, 1948
  - descriptor, 1949
  - err, 1949
  - FileDescriptor, 1948
  - in, 1949
  - out, 1949
  - readonly, 1949
  - sync, 1948
  - valid, 1948
- decaf::io::FilterInputStream, 1950
  - ~FilterInputStream, 1953
  - available, 1953
  - close, 1954
  - closed, 1957
  - doReadArray, 1954
  - doReadArrayBounded, 1954
  - doReadByte, 1954
  - FilterInputStream, 1953
  - inputStream, 1957
  - isClosed, 1955
  - mark, 1955
  - markSupported, 1955
  - own, 1957
  - reset, 1955
  - skip, 1956
- decaf::io::FilterOutputStream, 1957
  - ~FilterOutputStream, 1959
  - close, 1959
  - closed, 1961
  - doWriteArray, 1959
  - doWriteArrayBounded, 1960
  - doWriteByte, 1960
  - FilterOutputStream, 1959
  - flush, 1960
  - isClosed, 1960
  - outputStream, 1961
  - own, 1961
  - toString, 1961
- decaf::io::Flushable, 1998
  - ~Flushable, 1998
  - flush, 1998
- decaf::io::InputStream, 2105
  - ~InputStream, 2107
  - available, 2107
  - close, 2108
  - doReadArray, 2108
  - doReadArrayBounded, 2108
  - doReadByte, 2108
  - InputStream, 2107
  - lock, 2109



- mark, 2109
- markSupported, 2109
- notify, 2110
- notifyAll, 2110
- read, 2110–2112
- reset, 2113
- skip, 2113
- toString, 2114
- tryLock, 2114
- unlock, 2114
- wait, 2115, 2116
- decaf::io::InputStreamReader, 2116
  - ~InputStreamReader, 2118
  - checkClosed, 2118
  - close, 2118
  - doReadArrayBounded, 2118
  - InputStreamReader, 2117
  - ready, 2118
- decaf::io::InterruptedIOException, 2196
  - ~InterruptedIOException, 2198
  - clone, 2198
  - InterruptedIOException, 2197, 2198
- decaf::io::IOException, 2210
  - ~IOException, 2212
  - clone, 2212
  - IOException, 2211, 2212
- decaf::io::OutputStream, 2991
  - ~OutputStream, 2993
  - close, 2993
  - doWriteArray, 2993
  - doWriteArrayBounded, 2994
  - doWriteByte, 2994
  - flush, 2994
  - lock, 2994
  - notify, 2995
  - notifyAll, 2995
  - OutputStream, 2993
  - toString, 2995
  - tryLock, 2996
  - unlock, 2996
  - wait, 2996, 2997
  - write, 2998, 2999
- decaf::io::OutputStreamWriter, 2999
  - ~OutputStreamWriter, 3001
  - checkClosed, 3001
  - close, 3001
  - doWriteArrayBounded, 3001
  - flush, 3001
  - OutputStreamWriter, 3000
- decaf::io::PushbackInputStream, 3231
  - ~PushbackInputStream, 3234
  - available, 3234
  - doReadArrayBounded, 3235
  - doReadByte, 3235
  - mark, 3235
  - markSupported, 3235
  - PushbackInputStream, 3233, 3234
  - reset, 3236
  - skip, 3236
  - unread, 3237, 3238
- decaf::io::Reader, 3254
  - ~Reader, 3255
  - doReadArray, 3255
  - doReadArrayBounded, 3255
  - doReadChar, 3256
  - doReadCharBuffer, 3256
  - doReadVector, 3256
  - mark, 3256
  - markSupported, 3257
  - read, 3257–3259
  - Reader, 3255
  - ready, 3259
  - reset, 3260
  - skip, 3260
- decaf::io::UnsupportedEncodingException, 4025
  - ~UnsupportedEncodingException, 4027
  - clone, 4027
  - UnsupportedEncodingException, 4026, 4027
- decaf::io::UTFDataFormatException, 4078
  - ~UTFDataFormatException, 4080
  - clone, 4080
  - UTFDataFormatException, 4079, 4080
- decaf::io::Writer, 4133
  - ~Writer, 4135
  - append, 4135, 4136
  - doAppendChar, 4136
  - doAppendCharSequence, 4137
  - doAppendCharSequenceStartEnd, 4137
  - doWriteArray, 4137
  - doWriteArrayBounded, 4137
  - doWriteChar, 4137
  - doWriteString, 4137
  - doWriteStringBounded, 4137
  - doWriteVector, 4137
  - write, 4137–4139
  - Writer, 4135
- decaf::lang, 141
  - operator==, 144

- decaf::lang::Appendable, 733
  - ~Appendable, 734
  - append, 734
- decaf::lang::ArrayPointer, 736
  - ~ArrayPointer, 740
  - ArrayPointer, 739, 740
  - clone, 740
  - ConstReferenceType, 739
  - CounterType, 739
  - get, 740
  - length, 741
  - operator=, 742
  - operator==, 742, 744
  - operator[], 742
  - PointerType, 739
  - ReferenceType, 739
  - release, 743
  - reset, 743
  - swap, 743
- decaf::lang::ArrayPointerComparator, 744
  - compare, 745
  - operator(), 745
- decaf::lang::Boolean, 856
  - ~Boolean, 857
  - \_FALSE, 861
  - \_TRUE, 861
  - Boolean, 857
  - booleanValue, 857
  - compareTo, 857, 858
  - equals, 858
  - operator<, 858, 859
  - operator==, 859
  - parseBoolean, 860
  - toString, 860
  - valueOf, 860
- decaf::lang::Byte, 969
  - ~Byte, 972
  - Byte, 971
  - byteValue, 972
  - compareTo, 972
  - decode, 973
  - doubleValue, 973
  - equals, 973, 974
  - floatValue, 974
  - intValue, 974
  - longValue, 974
  - MAX\_VALUE, 979
  - MIN\_VALUE, 979
  - operator<, 974, 975
  - operator==, 975
  - parseByte, 976
  - shortValue, 977
  - SIZE, 979
  - toString, 977
  - valueOf, 977, 978
- decaf::lang::Character, 1126
  - byteValue, 1129
  - Character, 1128
  - compareTo, 1129
  - digit, 1129
  - doubleValue, 1130
  - equals, 1130
  - floatValue, 1130
  - intValue, 1131
  - isDigit, 1131
  - isISOControl, 1131
  - isLetter, 1131
  - isLetterOrDigit, 1131
  - isLowerCase, 1131
  - isUpperCase, 1132
  - isWhitespace, 1132
  - longValue, 1132
  - MAX\_RADIX, 1134
  - MAX\_VALUE, 1134
  - MIN\_RADIX, 1134
  - MIN\_VALUE, 1134
  - operator<, 1132
  - operator==, 1133
  - shortValue, 1133
  - SIZE, 1135
  - toString, 1134
  - valueOf, 1134
- decaf::lang::CharSequence, 1167
  - ~CharSequence, 1168
  - charAt, 1168
  - length, 1168
  - subSequence, 1168
  - toString, 1169
- decaf::lang::Comparable, 1248
  - ~Comparable, 1249
  - compareTo, 1249
  - equals, 1249
  - operator<, 1250
  - operator==, 1250
- decaf::lang::Double, 1841
  - ~Double, 1844
  - byteValue, 1844
  - compare, 1844
  - compareTo, 1844, 1845
  - Double, 1844

- doubleToLongBits, 1845
- doubleToRawLongBits, 1846
- doubleValue, 1846
- equals, 1846, 1847
- floatValue, 1847
- intValue, 1847
- isInfinite, 1847
- isNaN, 1848
- longBitsToDouble, 1848
- longValue, 1848
- MAX\_VALUE, 1853
- MIN\_VALUE, 1853
- NaN, 1853
- NEGATIVE\_INFINITY, 1853
- operator<, 1849
- operator==, 1849, 1850
- parseDouble, 1850
- POSITIVE\_INFINITY, 1853
- shortValue, 1850
- SIZE, 1853
- toHexString, 1851
- toString, 1851
- valueOf, 1852
- decaf::lang::DYNAMIC\_CAST\_TOKEN, 1878
- decaf::lang::Exception, 1886
  - ~Exception, 1889
  - buildMessage, 1889
  - cause, 1893
  - clone, 1889
  - Exception, 1888, 1889
  - getCause, 1890
  - getMessage, 1890
  - getStackTrace, 1891
  - getStackTraceString, 1891
  - initCause, 1891
  - message, 1893
  - operator=, 1892
  - printStackTrace, 1892
  - setMark, 1892
  - setMessage, 1892
  - setStackTrace, 1893
  - stackTrace, 1893
  - what, 1893
- decaf::lang::exceptions, 144
- decaf::lang::exceptions::ClassCastException, 1177
  - ~ClassCastException, 1179
  - ClassCastException, 1177, 1178
  - clone, 1179
- decaf::lang::exceptions::IllegalArgumentException, 2054
  - ~IllegalArgumentException, 2056
  - clone, 2056
  - IllegalArgumentException, 2055, 2056
- decaf::lang::exceptions::IllegalMonitorStateException, 2056
  - ~IllegalMonitorStateException, 2059
  - clone, 2059
  - IllegalMonitorStateException, 2057, 2058
- decaf::lang::exceptions::IllegalStateException, 2060
  - ~IllegalStateException, 2062
  - clone, 2062
  - IllegalStateException, 2061, 2062
- decaf::lang::exceptions::IllegalThreadStateException, 2063
  - ~IllegalThreadStateException, 2065
  - clone, 2065
  - IllegalThreadStateException, 2064, 2065
- decaf::lang::exceptions::IndexOutOfBoundsException, 2069
  - ~IndexOutOfBoundsException, 2071
  - clone, 2072
  - IndexOutOfBoundsException, 2070, 2071
- decaf::lang::exceptions::InterruptedException, 2193
  - ~InterruptedException, 2196
  - clone, 2196
  - InterruptedException, 2194, 2195
- decaf::lang::exceptions::InvalidStateException, 2207
  - ~InvalidStateException, 2210
  - clone, 2210
  - InvalidStateException, 2208, 2209
- decaf::lang::exceptions::NoSuchElementException, 2910
  - ~NoSuchElementException, 2912
  - clone, 2912
  - NoSuchElementException, 2911, 2912
- decaf::lang::exceptions::NullPointerException, 2915
  - ~NullPointerException, 2917
  - clone, 2918
  - NullPointerException, 2916, 2917
- decaf::lang::exceptions::NumberFormatException, 2921
  - ~NumberFormatException, 2923

- clone, 2923
- NumberFormatException, 2922, 2923
- decaf::lang::exceptions::RuntimeException, 3421
- ~RuntimeException, 3423
- clone, 3423
- RuntimeException, 3422, 3423
- decaf::lang::exceptions::UnsupportedOperationException, 4028
- ~UnsupportedOperationException, 4030
- clone, 4030
- UnsupportedOperationException, 4028, 4029
- decaf::lang::Float, 1961
- ~Float, 1964
- byteValue, 1964
- compare, 1965
- compareTo, 1965
- doubleValue, 1965
- equals, 1966
- Float, 1964
- floatToIntBits, 1966
- floatToRawIntBits, 1967
- floatValue, 1967
- intBitsToFloat, 1967
- intValue, 1968
- isInfinite, 1968
- isNaN, 1968
- longValue, 1969
- MAX\_VALUE, 1973
- MIN\_VALUE, 1973
- NaN, 1973
- NEGATIVE\_INFINITY, 1973
- operator<, 1969
- operator==, 1970
- parseFloat, 1970
- POSITIVE\_INFINITY, 1973
- shortValue, 1971
- SIZE, 1973
- toHexString, 1971
- toString, 1971, 1972
- valueOf, 1972, 1973
- decaf::lang::Integer, 2143
- ~Integer, 2146
- bitCount, 2147
- byteValue, 2147
- compareTo, 2147
- decode, 2148
- doubleValue, 2148
- equals, 2149
- floatValue, 2149
- highestOneBit, 2149
- Integer, 2146
- intValue, 2150
- longValue, 2150
- lowestOneBit, 2150
- MAX\_VALUE, 2159
- MIN\_VALUE, 2159
- numberOfLeadingZeros, 2150
- numberOfTrailingZeros, 2151
- operator<, 2151, 2152
- operator==, 2152
- parseInt, 2153
- reverse, 2154
- reverseBytes, 2154
- rotateLeft, 2154
- rotateRight, 2155
- shortValue, 2155
- signum, 2155
- SIZE, 2159
- toBinaryString, 2156
- toHexString, 2156
- toOctalString, 2156
- toString, 2157, 2158
- valueOf, 2158, 2159
- decaf::lang::Iterable, 2220
- ~Iterable, 2221
- iterator, 2221, 2222
- decaf::lang::Long, 2495
- ~Long, 2498
- bitCount, 2498
- byteValue, 2499
- compareTo, 2499
- decode, 2499
- doubleValue, 2500
- equals, 2500
- floatValue, 2501
- highestOneBit, 2501
- intValue, 2501
- Long, 2498
- longValue, 2501
- lowestOneBit, 2502
- MAX\_VALUE, 2510
- MIN\_VALUE, 2510
- numberOfLeadingZeros, 2502
- numberOfTrailingZeros, 2503
- operator<, 2503
- operator==, 2504
- parseLong, 2504, 2505
- reverse, 2505

- reverseBytes, 2505
- rotateLeft, 2506
- rotateRight, 2506
- shortValue, 2507
- signum, 2507
- SIZE, 2510
- toBinaryString, 2507
- toHexString, 2508
- toOctalString, 2508
- toString, 2508, 2509
- valueOf, 2509, 2510
- decaf::lang::Math, 2575
  - ~Math, 2578
  - abs, 2578, 2579
  - ceil, 2579
  - E, 2592
  - floor, 2580
  - Math, 2578
  - max, 2581, 2582
  - min, 2582–2584
  - PI, 2592
  - pow, 2585
  - random, 2585
  - round, 2586
  - signum, 2587
  - sqrt, 2588
  - toDegrees, 2591
  - toRadians, 2592
- decaf::lang::Number, 2918
  - ~Number, 2919
  - byteValue, 2919
  - doubleValue, 2919
  - floatValue, 2919
  - intValue, 2920
  - longValue, 2920
  - shortValue, 2920
- decaf::lang::Pointer, 3032
  - ~Pointer, 3036
  - CounterType, 3035
  - dynamicCast, 3037
  - get, 3037
  - operator\*, 3037
  - operator->, 3038
  - operator=, 3038
  - operator==, 3039, 3040
  - Pointer, 3035, 3036
  - PointerType, 3035
  - ReferenceType, 3035
  - release, 3039
  - reset, 3039
  - staticCast, 3039
  - swap, 3040
- decaf::lang::PointerComparator, 3040
  - compare, 3041
  - operator(), 3041
- decaf::lang::Readable, 3251
  - ~Readable, 3252
  - read, 3252
- decaf::lang::Runnable, 3418
  - ~Runnable, 3419
  - run, 3419
- decaf::lang::Runtime, 3419
  - ~Runtime, 3420
  - decaf::lang::System, 3846
  - decaf::lang::Thread, 3886
  - decaf::util::logging::LogManager, 2487
  - getRuntime, 3420
  - initializeRuntime, 3420
  - shutdownRuntime, 3421
- decaf::lang::Short, 3538
  - ~Short, 3541
  - byteValue, 3541
  - compareTo, 3541, 3542
  - decode, 3542
  - doubleValue, 3542
  - equals, 3543
  - floatValue, 3543
  - intValue, 3543
  - longValue, 3543
  - MAX\_VALUE, 3548
  - MIN\_VALUE, 3548
  - operator<, 3544
  - operator==, 3544, 3545
  - parseShort, 3545, 3546
  - reverseBytes, 3546
  - Short, 3541
  - shortValue, 3546
  - SIZE, 3548
  - toString, 3546, 3547
  - valueOf, 3547
- decaf::lang::STATIC\_CAST\_TOKEN, 3692
- decaf::lang::String, 3775
  - ~String, 3777
  - charAt, 3777
  - isEmpty, 3777
  - length, 3778
  - String, 3777
  - subSequence, 3778
  - toString, 3778
- decaf::lang::System, 3838

- ~System, 3840
- arraycopy, 3840, 3841
- availableProcessors, 3842
- clearProperty, 3842
- currentTimeMillis, 3843
- decaf::lang::Runtime, 3846
- getenv, 3843
- getProperties, 3843
- getProperty, 3844
- nanoTime, 3845
- setenv, 3845
- setProperty, 3845
- System, 3840
- unsetenv, 3846
- decaf::lang::Thread, 3876
  - ~Thread, 3881
  - BLOCKED, 3880
  - currentThread, 3881
  - decaf::lang::Runtime, 3886
  - decaf::util::concurrent::locks::LockSupport, clone, 844
  - 3886
  - getId, 3881
  - getName, 3882
  - getPriority, 3882
  - getState, 3882
  - getUncaughtExceptionHandler, 3882
  - isAlive, 3882
  - join, 3883
  - MAX\_PRIORITY, 3886
  - MIN\_PRIORITY, 3886
  - NEW, 3880
  - NORM\_PRIORITY, 3886
  - run, 3884
  - RUNNABLE, 3880
  - setName, 3884
  - setPriority, 3884
  - setUncaughtExceptionHandler, 3884
  - sleep, 3885
  - SLEEPING, 3880
  - start, 3885
  - State, 3880
  - TERMINATED, 3880
  - Thread, 3880, 3881
  - TIMED\_WAITING, 3880
  - toString, 3886
  - WAITING, 3880
  - yield, 3886
- decaf::lang::Thread::UncaughtExceptionHandler, ~InetAddress, 2080
  - 4018
  - ~UncaughtExceptionHandler, 4019
- uncaughtException, 4019
- decaf::lang::ThreadGroup, 3888
  - ~ThreadGroup, 3888
  - ThreadGroup, 3888
- decaf::lang::Throwable, 3895
  - ~Throwable, 3896
  - clone, 3896
  - getCause, 3897
  - getMessage, 3897
  - getStackTrace, 3897
  - getStackTraceString, 3898
  - initCause, 3898
  - printStackTrace, 3898
  - setMark, 3898
  - Throwable, 3896
- decaf::net, 145
- decaf::net::BindException, 842
  - ~BindException, 844
  - BindException, 843, 844
- decaf::net::ConnectException, 1293
  - ~ConnectException, 1295
  - clone, 1296
  - ConnectException, 1294, 1295
- decaf::net::HttpRetryException, 2049
  - ~HttpRetryException, 2051
  - clone, 2051
  - HttpRetryException, 2050, 2051
- decaf::net::Inet4Address, 2072
  - ~Inet4Address, 2073
  - Inet4Address, 2073
  - InetAddress, 2076
  - isAnyLocalAddress, 2073
  - isLinkLocalAddress, 2074
  - isLoopbackAddress, 2074
  - isMCGlobal, 2074
  - isMCLinkLocal, 2074
  - isMCNodeLocal, 2074
  - isMCOrgLocal, 2075
  - isMCSiteLocal, 2075
  - isMulticastAddress, 2075
  - isSiteLocalAddress, 2075
- decaf::net::Inet6Address, 2076
  - ~Inet6Address, 2077
  - Inet6Address, 2077
  - InetAddress, 2077
- decaf::net::InetAddress, 2077
  - ~InetAddress, 2080
  - addressBytes, 2085
  - anyBytes, 2085

- bytesToInt, 2080
- getAddress, 2080
- getAnyAddress, 2080
- getByAddress, 2080, 2081
- getHostAddress, 2081
- getHostName, 2081
- getLocalHost, 2082
- getLoopbackAddress, 2082
- hostname, 2085
- InetAddress, 2080
- isAnyLocalAddress, 2082
- isLinkLocalAddress, 2082
- isLoopbackAddress, 2082
- isMCGlobal, 2083
- isMCLinkLocal, 2083
- isMCNodeLocal, 2083
- isMCOrgLocal, 2083
- isMCSiteLocal, 2084
- isMulticastAddress, 2084
- isSiteLocalAddress, 2084
- loopbackBytes, 2085
- reached, 2085
- toString, 2084
- decaf::net::InetSocketAddress, 2085
  - ~InetSocketAddress, 2085
  - InetSocketAddress, 2085
- decaf::net::MalformedURLException, 2536
  - ~MalformedURLException, 2538
  - clone, 2538
  - MalformedURLException, 2537, 2538
- decaf::net::NoRouteToHostException, 2905
  - ~NoRouteToHostException, 2907
  - clone, 2907
  - NoRouteToHostException, 2905, 2906
- decaf::net::PortUnreachableException, 3060
  - ~PortUnreachableException, 3062
  - clone, 3062
  - PortUnreachableException, 3061, 3062
- decaf::net::ProtocolException, 3228
  - ~ProtocolException, 3230
  - clone, 3230
  - ProtocolException, 3229, 3230
- decaf::net::ServerSocket, 3447
  - ~ServerSocket, 3451
  - accept, 3451
  - bind, 3452
  - checkClosed, 3453
  - close, 3453
  - ensureCreated, 3453
  - getDefaultBacklog, 3453
  - getLocalPort, 3453
  - getReceiveBufferSize, 3453
  - getReuseAddress, 3454
  - getSoTimeout, 3454
  - implAccept, 3454
  - isBound, 3454
  - isClosed, 3455
  - ServerSocket, 3449–3451
  - setReceiveBufferSize, 3455
  - setReuseAddress, 3455
  - setSocketImplFactory, 3455
  - setSoTimeout, 3456
  - setupSocketImpl, 3456
  - toString, 3456
- decaf::net::ServerSocketFactory, 3457
  - ~ServerSocketFactory, 3458
  - createServerSocket, 3458, 3459
  - getDefault, 3460
  - ServerSocketFactory, 3458
- decaf::net::Socket, 3607
  - ~Socket, 3614
  - accepted, 3614
  - bind, 3614
  - checkClosed, 3614
  - close, 3614
  - connect, 3614, 3615
  - ensureCreated, 3615
  - getInetAddress, 3616
  - getInputStream, 3616
  - getKeepAlive, 3616
  - getLocalAddress, 3617
  - getLocalPort, 3617
  - getOOBInline, 3617
  - getOutputStream, 3617
  - getPort, 3618
  - getReceiveBufferSize, 3618
  - getReuseAddress, 3618
  - getSendBufferSize, 3618
  - getSoLinger, 3619
  - getSoTimeout, 3619
  - getTcpNoDelay, 3619
  - getTrafficClass, 3620
  - impl, 3625
  - initSocketImpl, 3620
  - isBound, 3620
  - isClosed, 3620
  - isConnected, 3620
  - isInputShutdown, 3620
  - isOutputShutdown, 3621
  - sendUrgentData, 3621

- ServerSocket, 3625
- setKeepAlive, 3621
- setOOBInline, 3621
- setReceiveBufferSize, 3622
- setReuseAddress, 3622
- setSendBufferSize, 3622
- setSocketImplFactory, 3623
- setSoLinger, 3623
- setSoTimeout, 3623
- setTcpNoDelay, 3624
- setTrafficClass, 3624
- shutdownInput, 3625
- shutdownOutput, 3625
- Socket, 3611–3613
- toString, 3625
- decaf::net::SocketAddress, 3626
  - ~SocketAddress, 3626
- decaf::net::SocketError, 3626
  - getErrorCode, 3627
  - getErrorString, 3627
- decaf::net::SocketException, 3627
  - ~SocketException, 3629
  - clone, 3629
  - SocketException, 3628
- decaf::net::SocketFactory, 3629
  - ~SocketFactory, 3631
  - createSocket, 3631–3633
  - getDefault, 3633
  - SocketFactory, 3631
- decaf::net::SocketImpl, 3635
  - ~SocketImpl, 3638
  - accept, 3638
  - address, 3644
  - available, 3638
  - bind, 3638
  - close, 3639
  - connect, 3639
  - create, 3639
  - fd, 3644
  - getFileDescriptor, 3640
  - getInetAddress, 3640
  - getInputStream, 3640
  - getLocalAddress, 3640
  - getLocalPort, 3641
  - getOption, 3641
  - getOutputStream, 3641
  - getPort, 3642
  - listen, 3642
  - localPort, 3644
  - port, 3644
  - sendUrgentData, 3642
  - setOption, 3642
  - shutdownInput, 3643
  - shutdownOutput, 3643
  - SocketImpl, 3638
  - supportsUrgentData, 3643
  - toString, 3644
- decaf::net::SocketImplFactory, 3644
  - ~SocketImplFactory, 3645
  - createSocketImpl, 3645
- decaf::net::SocketOptions, 3645
  - ~SocketOptions, 3647
  - SOCKET\_OPTION\_BINDADDR, 3647
  - SOCKET\_OPTION\_BROADCAST, 3647
  - SOCKET\_OPTION\_IP\_MULTICAST - IF, 3647
  - SOCKET\_OPTION\_IP\_MULTICAST - IF2, 3648
  - SOCKET\_OPTION\_IP\_MULTICAST - LOOP, 3648
  - SOCKET\_OPTION\_IP\_TOS, 3648
  - SOCKET\_OPTION\_KEEPALIVE, 3648
  - SOCKET\_OPTION\_LINGER, 3648
  - SOCKET\_OPTION\_OOBLINE, 3649
  - SOCKET\_OPTION\_RCVBUF, 3649
  - SOCKET\_OPTION\_REUSEADDR, 3649
  - SOCKET\_OPTION\_SNDBUF, 3649
  - SOCKET\_OPTION\_TCP\_NODELAY, 3650
  - SOCKET\_OPTION\_TIMEOUT, 3650
- decaf::net::SocketTimeoutException, 3650
  - ~SocketTimeoutException, 3652
  - clone, 3652
  - SocketTimeoutException, 3651, 3652
- decaf::net::ssl, 146
- decaf::net::ssl::SSLContext, 3653
  - ~SSLContext, 3654
  - getDefault, 3654
  - getDefaultSSLParameters, 3654
  - getServerSocketFactory, 3654
  - getSocketFactory, 3654
  - getSupportedSSLParameters, 3655
  - setDefault, 3655
  - SSLContext, 3654
- decaf::net::ssl::SSLContextSpi, 3655
  - ~SSLContextSpi, 3656
  - providerGetDefaultSSLParameters, 3656
  - providerGetServerSocketFactory, 3657



- providerGetSocketFactory, 3657
- providerGetSupportedSSLParameters, 3658
- providerInit, 3658
- decaf::net::ssl::SSLParameters, 3658
  - ~SSLParameters, 3660
  - getCipherSuites, 3660
  - getNeedClientAuth, 3660
  - getProtocols, 3660
  - getWantClientAuth, 3660
  - setCipherSuites, 3660
  - setNeedClientAuth, 3661
  - setProtocols, 3661
  - setWantClientAuth, 3661
  - SSLParameters, 3659, 3660
- decaf::net::ssl::SSLServerSocket, 3662
  - ~SSLServerSocket, 3665
  - getEnabledCipherSuites, 3665
  - getEnabledProtocols, 3665
  - getNeedClientAuth, 3665
  - getSupportedCipherSuites, 3665
  - getSupportedProtocols, 3666
  - getWantClientAuth, 3666
  - setEnabledCipherSuites, 3666
  - setEnabledProtocols, 3667
  - setNeedClientAuth, 3667
  - setWantClientAuth, 3667
  - SSLServerSocket, 3663, 3664
- decaf::net::ssl::SSLServerSocketFactory, 3668
  - ~SSLServerSocketFactory, 3669
  - getDefault, 3669
  - getDefaultCipherSuites, 3669
  - getSupportedCipherSuites, 3669
  - SSLServerSocketFactory, 3669
- decaf::net::ssl::SSLSocket, 3670
  - ~SSLSocket, 3674
  - getEnabledCipherSuites, 3674
  - getEnabledProtocols, 3674
  - getNeedClientAuth, 3675
  - getSSLParameters, 3675
  - getSupportedCipherSuites, 3675
  - getSupportedProtocols, 3675
  - getUseClientMode, 3676
  - getWantClientAuth, 3676
  - setEnabledCipherSuites, 3676
  - setEnabledProtocols, 3677
  - setNeedClientAuth, 3677
  - setSSLParameters, 3677
  - setUseClientMode, 3678
  - setWantClientAuth, 3678
  - SSLSocket, 3672, 3673
  - startHandshake, 3678
- decaf::net::ssl::SSLSocketFactory, 3679
  - ~SSLSocketFactory, 3680
  - createSocket, 3680
  - getDefault, 3681
  - getDefaultCipherSuites, 3681
  - getSupportedCipherSuites, 3681
  - SSLSocketFactory, 3680
- decaf::net::UnknownHostException, 4019
  - ~UnknownHostException, 4022
  - clone, 4022
  - UnknownHostException, 4020, 4021
- decaf::net::UnknownServiceException, 4022
  - ~UnknownServiceException, 4024
  - clone, 4024
  - UnknownServiceException, 4023, 4024
- decaf::net::URI, 4031
  - ~URI, 4036
  - compareTo, 4036
  - create, 4036
  - equals, 4037
  - getAuthority, 4037
  - getFragment, 4037
  - getHost, 4037
  - getPath, 4037
  - getPort, 4037
  - getQuery, 4037
  - getRawAuthority, 4037
  - getRawFragment, 4038
  - getRawPath, 4038
  - getRawQuery, 4038
  - getRawSchemeSpecificPart, 4038
  - getRawUserInfo, 4039
  - getScheme, 4039
  - getSchemeSpecificPart, 4039
  - getUserInfo, 4039
  - isAbsolute, 4039
  - isOpaque, 4040
  - normalize, 4040
  - operator<, 4040
  - operator==, 4041
  - parseServerAuthority, 4041
  - relativize, 4041
  - resolve, 4042
  - toString, 4043
  - toURL, 4043
  - URI, 4034, 4035
- decaf::net::URISyntaxException, 4060
  - ~URISyntaxException, 4063

- clone, 4063
- getIndex, 4063
- getInput, 4063
- getReason, 4063
- URISyntaxException, 4061, 4062
- decaf::net::URL, 4072
  - ~URL, 4073
  - URL, 4073
- decaf::net::URLDecoder, 4074
  - ~URLDecoder, 4074
  - decode, 4074
- decaf::net::URLEncoder, 4074
  - ~URLEncoder, 4075
  - encode, 4075
- decaf::nio, 147
- decaf::nio::Buffer, 936
  - ~Buffer, 939
  - \_capacity, 942
  - \_limit, 942
  - \_mark, 942
  - \_markSet, 942
  - \_position, 942
  - Buffer, 939
  - capacity, 939
  - clear, 939
  - flip, 939
  - hasRemaining, 940
  - isReadOnly, 940
  - limit, 940
  - mark, 941
  - position, 941
  - remaining, 941
  - reset, 941
  - rewind, 942
- decaf::nio::BufferOverflowException, 964
  - ~BufferOverflowException, 966
  - BufferOverflowException, 965, 966
  - clone, 966
- decaf::nio::BufferUnderflowException, 967
  - ~BufferUnderflowException, 969
  - BufferUnderflowException, 967, 968
  - clone, 969
- decaf::nio::ByteBuffer, 1049
  - ~ByteBuffer, 1056
  - allocate, 1056
  - array, 1056
  - arrayOffset, 1057
  - asCharBuffer, 1057
  - asDoubleBuffer, 1057
  - asFloatBuffer, 1058
  - asIntBuffer, 1058
  - asLongBuffer, 1059
  - asReadOnlyBuffer, 1059
  - asShortBuffer, 1059
  - ByteBuffer, 1055
  - compact, 1060
  - compareTo, 1060
  - duplicate, 1060
  - equals, 1061
  - get, 1061, 1062
  - getChar, 1063
  - getDouble, 1063, 1064
  - getFloat, 1064, 1065
  - getInt, 1065
  - getLong, 1066
  - getShort, 1067
  - hasArray, 1067
  - isReadOnly, 1068
  - operator<, 1068
  - operator==, 1068
  - put, 1068–1071
  - putChar, 1071, 1072
  - putDouble, 1072, 1073
  - putFloat, 1073, 1074
  - putInt, 1074, 1075
  - putLong, 1075, 1076
  - putShort, 1076, 1077
  - slice, 1077
  - toString, 1078
  - wrap, 1078
- decaf::nio::CharBuffer, 1148
  - ~CharBuffer, 1152
  - allocate, 1152
  - append, 1152–1154
  - array, 1154
  - arrayOffset, 1155
  - asReadOnlyBuffer, 1155
  - charAt, 1155
  - CharBuffer, 1152
  - compact, 1156
  - compareTo, 1156
  - duplicate, 1157
  - equals, 1157
  - get, 1157, 1158
  - hasArray, 1159
  - length, 1159
  - operator<, 1159
  - operator==, 1160
  - put, 1160–1163
  - read, 1164

- slice, 1164
- subSequence, 1165
- toString, 1165
- wrap, 1166
- decaf::nio::DoubleBuffer, 1865
  - ~DoubleBuffer, 1868
  - allocate, 1868
  - array, 1868
  - arrayOffset, 1869
  - asReadOnlyBuffer, 1869
  - compact, 1870
  - compareTo, 1870
  - DoubleBuffer, 1868
  - duplicate, 1870
  - equals, 1871
  - get, 1871, 1872
  - hasArray, 1873
  - operator<, 1873
  - operator==, 1873
  - put, 1873–1875
  - slice, 1876
  - toString, 1876
  - wrap, 1877
- decaf::nio::FloatBuffer, 1985
  - ~FloatBuffer, 1988
  - allocate, 1988
  - array, 1988
  - arrayOffset, 1989
  - asReadOnlyBuffer, 1989
  - compact, 1990
  - compareTo, 1990
  - duplicate, 1991
  - equals, 1991
  - FloatBuffer, 1988
  - get, 1991, 1992
  - hasArray, 1993
  - operator<, 1993
  - operator==, 1993
  - put, 1993–1996
  - slice, 1996
  - toString, 1996
  - wrap, 1997
- decaf::nio::IntBuffer, 2130
  - ~IntBuffer, 2133
  - allocate, 2133
  - array, 2133
  - arrayOffset, 2134
  - asReadOnlyBuffer, 2134
  - compact, 2135
  - compareTo, 2135
  - duplicate, 2136
  - equals, 2136
  - get, 2136, 2137
  - hasArray, 2138
  - IntBuffer, 2133
  - operator<, 2138
  - operator==, 2138
  - put, 2138–2141
  - slice, 2141
  - toString, 2141
  - wrap, 2142
- decaf::nio::InvalidMarkException, 2204
  - ~InvalidMarkException, 2206
  - clone, 2206
  - InvalidMarkException, 2204, 2205
- decaf::nio::LongBuffer, 2522
  - ~LongBuffer, 2525
  - allocate, 2525
  - array, 2525
  - arrayOffset, 2526
  - asReadOnlyBuffer, 2526
  - compact, 2527
  - compareTo, 2527
  - duplicate, 2528
  - equals, 2528
  - get, 2528, 2529
  - hasArray, 2530
  - LongBuffer, 2525
  - operator<, 2530
  - operator==, 2530
  - put, 2530–2533
  - slice, 2533
  - toString, 2533
  - wrap, 2534
- decaf::nio::ReadOnlyBufferException, 3260
  - ~ReadOnlyBufferException, 3262
  - clone, 3263
  - ReadOnlyBufferException, 3261, 3262
- decaf::nio::ShortBuffer, 3560
  - ~ShortBuffer, 3563
  - allocate, 3563
  - array, 3563
  - arrayOffset, 3564
  - asReadOnlyBuffer, 3564
  - compact, 3565
  - compareTo, 3565
  - duplicate, 3565
  - equals, 3566
  - get, 3566, 3567
  - hasArray, 3568

- operator<, 3568
- operator==, 3568
- put, 3568–3570
- ShortBuffer, 3563
- slice, 3571
- toString, 3571
- wrap, 3571, 3572
- decaf::security, 147
- decaf::security::auth, 148
- decaf::security::auth::x500, 148
- decaf::security::auth::x500::X500Principal, 4140
  - ~X500Principal, 4140
  - getEncoded, 4140
  - getName, 4140
  - hashCode, 4140
- decaf::security::cert, 148
- decaf::security::cert::Certificate, 1112
  - ~Certificate, 1113
  - equals, 1113
  - getEncoded, 1114
  - getPublicKey, 1114
  - getType, 1114
  - toString, 1114
  - verify, 1115
- decaf::security::cert::CertificateEncodingException, 1116
  - ~CertificateEncodingException, 1117
  - CertificateEncodingException, 1117
  - clone, 1118
- decaf::security::cert::CertificateException, 1118
  - ~CertificateException, 1119
  - CertificateException, 1119
  - clone, 1120
- decaf::security::cert::CertificateExpiredException, 1120
  - ~CertificateExpiredException, 1122
  - CertificateExpiredException, 1121
  - clone, 1122
- decaf::security::cert::CertificateNotYetValidException, 1122
  - ~CertificateNotYetValidException, 1124
  - CertificateNotYetValidException, 1123
  - clone, 1124
- decaf::security::cert::CertificateParsingException, 1124
  - ~CertificateParsingException, 1126
  - CertificateParsingException, 1125
  - clone, 1126
- decaf::security::cert::X509Certificate, 4141
  - ~X509Certificate, 4142
  - checkValidity, 4142
  - getBasicConstraints, 4142
  - getIssuerUniqueID, 4142
  - getIssuerX500Principal, 4142
  - getKeyUsage, 4142
  - getNotAfter, 4142
  - getNotBefore, 4142
  - getSigAlgName, 4142
  - getSigAlgOID, 4142
  - getSigAlgParams, 4142
  - getSignature, 4142
  - getSubjectUniqueID, 4142
  - getSubjectX500Principal, 4142
  - getTBSCertificate, 4142
  - getVersion, 4142
- decaf::security::GeneralSecurityException, 2034
  - ~GeneralSecurityException, 2036
  - clone, 2037
  - GeneralSecurityException, 2035, 2036
- decaf::security::InvalidKeyException, 2201
  - ~InvalidKeyException, 2203
  - clone, 2203
- decaf::security::InvalidKeyException, 2202, 2203
- decaf::security::Key, 2364
  - ~Key, 2365
  - getAlgorithm, 2365
  - getEncoded, 2365
  - getFormat, 2365
- decaf::security::KeyException, 2366
  - ~KeyException, 2368
  - clone, 2368
  - KeyException, 2367, 2368
- decaf::security::KeyManagementException, 2368
  - ~KeyManagementException, 2371
  - clone, 2371
  - KeyManagementException, 2369, 2370
- decaf::security::NoSuchAlgorithmException, 2907
  - ~NoSuchAlgorithmException, 2909
  - clone, 2910
  - NoSuchAlgorithmException, 2908, 2909
- decaf::security::NoSuchProviderException, 2913
  - ~NoSuchProviderException, 2915
  - clone, 2915
  - NoSuchProviderException, 2913, 2914

- decaf::security::Principal, 3114
  - ~Principal, 3115
  - equals, 3115
  - getName, 3115
- decaf::security::PublicKey, 3231
  - ~PublicKey, 3231
- decaf::security::SecureRandom, 3424
  - ~SecureRandom, 3426
  - next, 3427
  - nextBytes, 3427, 3428
  - SecureRandom, 3426
  - setSeed, 3428, 3429
- decaf::security::SecureRandomSpi, 3432
  - ~SecureRandomSpi, 3433
  - providerGenerateSeed, 3433
  - providerNextBytes, 3433
  - providerSetSeed, 3434
  - SecureRandomSpi, 3433
- decaf::security::SignatureException, 3601
  - ~SignatureException, 3603
  - clone, 3604
  - SignatureException, 3602, 3603
- decaf::util, 149
- decaf::util::AbstractCollection, 159
  - ~AbstractCollection, 162
  - AbstractCollection, 162
  - add, 162
  - addAll, 163
  - clear, 163
  - contains, 164
  - containsAll, 165
  - copy, 165
  - equals, 166
  - isEmpty, 166
  - lock, 166
  - mutex, 173
  - notify, 167
  - notifyAll, 167
  - operator=, 167
  - remove, 168
  - removeAll, 169
  - retainAll, 169
  - toArray, 170
  - tryLock, 171
  - unlock, 171
  - wait, 171, 172
- decaf::util::AbstractList, 173
  - ~AbstractList, 174
- decaf::util::AbstractMap, 174
  - ~AbstractMap, 175
- decaf::util::AbstractQueue, 175
  - ~AbstractQueue, 177
  - AbstractQueue, 177
  - add, 177
  - addAll, 177
  - clear, 178
  - element, 178
  - remove, 179
- decaf::util::AbstractSequentialList, 179
  - ~AbstractSequentialList, 180
- decaf::util::AbstractSet, 180
  - ~AbstractSet, 181
  - removeAll, 181
- decaf::util::Collection, 1216
  - ~Collection, 1218
  - add, 1218
  - addAll, 1219
  - clear, 1220
  - contains, 1221
  - containsAll, 1221
  - equals, 1222
  - isEmpty, 1223
  - remove, 1223
  - removeAll, 1224
  - retainAll, 1225
  - size, 1226
  - toArray, 1226
- decaf::util::Comparator, 1251
  - ~Comparator, 1252
  - compare, 1252
  - operator(), 1252
- decaf::util::comparators, 151
- decaf::util::comparators::Less, 2399
  - ~Less, 2400
  - compare, 2400
  - Less, 2400
  - operator(), 2401
- decaf::util::concurrent, 151
- decaf::util::concurrent::atomic, 153
- decaf::util::concurrent::atomic::AtomicBoolean, 745
  - ~AtomicBoolean, 746
  - AtomicBoolean, 746
  - compareAndSet, 747
  - get, 747
  - getAndSet, 747
  - set, 747
  - toString, 748
- decaf::util::concurrent::atomic::AtomicInteger, 748

- ~AtomicInteger, 750
- addAndGet, 750
- AtomicInteger, 750
- compareAndSet, 750
- decrementAndGet, 750
- doubleValue, 751
- floatValue, 751
- get, 751
- getAndAdd, 751
- getAndDecrement, 752
- getAndIncrement, 752
- getAndSet, 752
- incrementAndGet, 752
- intValue, 752
- longValue, 753
- set, 753
- toString, 753
- decaf::util::concurrent::atomic::AtomicRefCounter, 754
  - ~AtomicRefCounter, 755
  - AtomicRefCounter, 755
  - release, 755
  - swap, 756
- decaf::util::concurrent::atomic::AtomicReference, 756
  - ~AtomicReference, 757
  - AtomicReference, 757
  - compareAndSet, 757
  - get, 757
  - getAndSet, 758
  - set, 758
  - toString, 758
- decaf::util::concurrent::BlockingQueue, 848
  - ~BlockingQueue, 852
  - drainTo, 852
  - offer, 853
  - poll, 854
  - put, 854
  - remainingCapacity, 855
  - take, 855
- decaf::util::concurrent::BrokenBarrierException, 866
  - ~BrokenBarrierException, 868
  - BrokenBarrierException, 867, 868
  - clone, 868
- decaf::util::concurrent::Callable, 1108
  - ~Callable, 1109
  - call, 1109
- decaf::util::concurrent::CancellationException, 1110
  - ~CancellationException, 1112
  - CancellationException, 1110, 1111
  - clone, 1112
- decaf::util::concurrent::ConcurrentMap, 1260
  - ~ConcurrentMap, 1261
  - putIfAbsent, 1261
  - remove, 1262
  - replace, 1263, 1264
- decaf::util::concurrent::ConcurrentStlMap, 1265
  - ~ConcurrentStlMap, 1270
  - clear, 1270
  - ConcurrentStlMap, 1269, 1270
  - containsKey, 1270
  - containsValue, 1271
  - copy, 1271
  - equals, 1272
  - get, 1272, 1273
  - isEmpty, 1273
  - keySet, 1273
  - lock, 1274
  - notify, 1274
  - notifyAll, 1274
  - put, 1275
  - putAll, 1275, 1276
  - putIfAbsent, 1276
  - remove, 1277
  - replace, 1278, 1279
  - size, 1279
  - tryLock, 1280
  - unlock, 1280
  - values, 1280
  - wait, 1280, 1281
- decaf::util::concurrent::ConditionHandle, 1290
  - ~ConditionHandle, 1291
  - condition, 1291
  - ConditionHandle, 1291
  - criticalSection, 1291
  - generation, 1291
  - mutex, 1291
  - numWaiting, 1291
  - numWake, 1291
  - semaphore, 1291
- decaf::util::concurrent::CountDownLatch, 1565
  - ~CountDownLatch, 1566
  - await, 1566, 1567
  - countDown, 1568
  - CountDownLatch, 1566
  - getCount, 1568

- decaf::util::concurrent::Delayed, 1774
  - ~Delayed, 1775
  - getDelay, 1775
- decaf::util::concurrent::ExecutionException, 1923
  - ~ExecutionException, 1925
  - clone, 1926
  - ExecutionException, 1924, 1925
- decaf::util::concurrent::Executor, 1926
  - ~Executor, 1928
  - execute, 1928
- decaf::util::concurrent::ExecutorService, 1928
  - ~ExecutorService, 1929
  - awaitTermination, 1929
- decaf::util::concurrent::Future, 2029
  - ~Future, 2030
  - cancel, 2030
  - get, 2031
  - isCancelled, 2032
  - isDone, 2032
- decaf::util::concurrent::Lock, 2450
  - ~Lock, 2451
  - isLocked, 2451
  - Lock, 2451
  - lock, 2451
  - unlock, 2451
- decaf::util::concurrent::locks, 153
- decaf::util::concurrent::locks::Condition, 1282
  - ~Condition, 1285
  - await, 1285, 1286
  - awaitNanos, 1286
  - awaitUninterruptibly, 1288
  - awaitUntil, 1289
  - signal, 1289
  - signalAll, 1289
- decaf::util::concurrent::locks::Lock, 2452
  - ~Lock, 2454
  - lock, 2454
  - lockInterruptibly, 2454
  - newCondition, 2455
  - tryLock, 2455, 2457
  - unlock, 2457
- decaf::util::concurrent::locks::LockSupport, 2458
  - ~LockSupport, 2459
- decaf::lang::Thread, 3886
  - park, 2459
  - parkNanos, 2460
  - parkUntil, 2460
  - unpark, 2461
- decaf::util::concurrent::locks::ReadWriteLock, 3263
  - ~ReadWriteLock, 3265
  - readLock, 3265
  - writeLock, 3265
- decaf::util::concurrent::locks::ReentrantLock, 3273
  - ~ReentrantLock, 3275
  - getHoldCount, 3275
  - isFair, 3275
  - isHeldByCurrentThread, 3275
  - isLocked, 3276
  - lock, 3276
  - lockInterruptibly, 3276
  - newCondition, 3277
  - ReentrantLock, 3275
  - toString, 3278
  - tryLock, 3278, 3279
  - unlock, 3279
- decaf::util::concurrent::Mutex, 2866
  - ~Mutex, 2868
  - lock, 2868
  - Mutex, 2868
  - notify, 2868
  - notifyAll, 2868
  - tryLock, 2869
  - unlock, 2869
  - wait, 2870
- decaf::util::concurrent::MutexHandle, 2871
  - ~MutexHandle, 2872
  - lock\_count, 2872
  - lock\_owner, 2872
  - mutex, 2872
  - MutexHandle, 2872
- decaf::util::concurrent::PooledThread, 3056
  - ~PooledThread, 3057
  - getPooledThreadListener, 3057
  - isBusy, 3057
  - PooledThread, 3057
  - run, 3057
  - setPooledThreadListener, 3057
  - stop, 3058
- decaf::util::concurrent::PooledThreadListener, 3058
  - ~PooledThreadListener, 3059
  - onTaskCompleted, 3059
  - onTaskException, 3059
  - onTaskStarted, 3059
- decaf::util::concurrent::RejectedExecutionException, 3280

- ~RejectedExecutionException, 3282
- clone, 3282
- RejectedExecutionException, 3281, 3282
- decaf::util::concurrent::RejectedExecutionHandler, 3283
  - ~RejectedExecutionHandler, 3283
  - rejectedExecution, 3283
- decaf::util::concurrent::Semaphore, 3434
  - ~Semaphore, 3438
  - acquire, 3438
  - acquireUninterruptibly, 3439, 3440
  - availablePermits, 3440
  - drainPermits, 3440
  - isFair, 3441
  - release, 3441
  - Semaphore, 3437
  - toString, 3442
  - tryAcquire, 3442–3444
- decaf::util::concurrent::Synchronizable, 3811
  - ~Synchronizable, 3812
  - lock, 3812
  - notify, 3813
  - notifyAll, 3814
  - tryLock, 3815
  - unlock, 3816
  - wait, 3817, 3819, 3820
- decaf::util::concurrent::SynchronousQueue, 3827
  - ~SynchronousQueue, 3830
  - clear, 3830
  - contains, 3830
  - containsAll, 3830
  - drainTo, 3831
  - equals, 3832
  - isEmpty, 3833
  - iterator, 3833
  - offer, 3833, 3834
  - peek, 3834
  - poll, 3834, 3835
  - put, 3835
  - remainingCapacity, 3836
  - remove, 3836
  - removeAll, 3837
  - retainAll, 3837
  - size, 3837
  - SynchronousQueue, 3830
  - take, 3837
  - toArray, 3838
- decaf::util::concurrent::TaskListener, 3847
  - ~TaskListener, 3848
  - onTaskComplete, 3848
  - onTaskException, 3848
- decaf::util::concurrent::ThreadFactory, 3887
  - newThread, 3887
- decaf::util::concurrent::ThreadPool, 3888
  - ~ThreadPool, 3891
  - DEFAULT\_MAX\_BLOCK\_SIZE, 3894
  - DEFAULT\_MAX\_POOL\_SIZE, 3894
  - deQueueTask, 3891
  - getBacklog, 3891
  - getBlockSize, 3891
  - getFreeThreadCount, 3892
  - getInstance, 3892
  - getMaxThreads, 3892
  - getPoolSize, 3892
  - onTaskCompleted, 3892
  - onTaskException, 3893
  - onTaskStarted, 3893
  - queueTask, 3893
  - reserve, 3893
  - setBlockSize, 3894
  - setMaxThreads, 3894
  - Task, 3891
  - ThreadPool, 3891
- decaf::util::concurrent::TimeoutException, 3899
  - ~TimeoutException, 3901
  - clone, 3901
  - TimeoutException, 3900, 3901
- decaf::util::concurrent::TimeUnit, 3919
  - ~TimeUnit, 3922
  - compareTo, 3922
  - convert, 3923
  - DAYS, 3929
  - equals, 3923
  - HOURS, 3929
  - MICROSECONDS, 3929
  - MILLISECONDS, 3929
  - MINUTES, 3929
  - NANOSECONDS, 3929
  - operator<, 3924
  - operator==, 3924
  - SECONDS, 3929
  - sleep, 3924
  - timedJoin, 3925
  - timedWait, 3925
  - TimeUnit, 3922
  - toDays, 3926
  - toHours, 3926



- toMicros, 3926
- toMillis, 3927
- toMinutes, 3927
- toNanos, 3928
- toSeconds, 3928
- toString, 3928
- valueOf, 3928
- values, 3930
- decaf::util::Date, 1719
  - ~Date, 1720
  - after, 1721
  - before, 1721
  - compareTo, 1721
  - Date, 1720
  - equals, 1721
  - getTime, 1721
  - operator<, 1722
  - operator=, 1722
  - operator==, 1722
  - setTime, 1722
  - toString, 1723
- decaf::util::Iterator, 2222
  - ~Iterator, 2223
  - hasNext, 2223
  - next, 2223
  - remove, 2223
- decaf::util::List, 2409
  - ~List, 2410
  - add, 2410
  - addAll, 2411
  - get, 2412
  - indexOf, 2412
  - lastIndexOf, 2413
  - List, 2410
  - listIterator, 2413, 2414
  - remove, 2415
  - set, 2416
- decaf::util::ListIterator, 2417
  - ~ListIterator, 2418
  - add, 2418
  - hasPrevious, 2418
  - nextIndex, 2418
  - previous, 2419
  - previousIndex, 2419
  - set, 2419
- decaf::util::logging, 154
  - Debug, 155
  - Error, 156
  - Fatal, 156
  - Info, 156
  - Levels, 155
  - Markblock, 155
  - Null, 155
  - Off, 155
  - Throwing, 156
  - Warn, 156
- decaf::util::logging::ConsoleHandler, 1439
  - ~ConsoleHandler, 1440
  - close, 1440
  - ConsoleHandler, 1440
  - publish, 1441
- decaf::util::logging::ErrorManager, 1884
  - ~ErrorManager, 1885
  - CLOSE\_FAILURE, 1885
  - error, 1885
  - ErrorManager, 1885
  - FLUSH\_FAILURE, 1885
  - FORMAT\_FAILURE, 1886
  - GENERIC\_FAILURE, 1886
  - OPEN\_FAILURE, 1886
  - WRITE\_FAILURE, 1886
- decaf::util::logging::Filter, 1949
  - ~Filter, 1950
  - isLoggable, 1950
- decaf::util::logging::Formatter, 2027
  - ~Formatter, 2028
  - format, 2028
  - formatMessage, 2028
  - getHead, 2028
  - getTail, 2029
- decaf::util::logging::Handler, 2042
  - ~Handler, 2043
  - flush, 2043
  - getErrorManager, 2043
  - getFilter, 2043
  - getFormatter, 2044
  - getLevel, 2044
  - Handler, 2043
  - isLoggable, 2044
  - publish, 2044
  - reportError, 2045
  - setErrorManager, 2045
  - setFilter, 2045
  - setFormatter, 2045
  - setLevel, 2046
- decaf::util::logging::Level, 2403
  - ~Level, 2405
  - ALL, 2407
  - compareTo, 2406
  - CONFIG, 2407

- DEBUG, 2407
- equals, 2406
- FINE, 2407
- FINER, 2408
- FINEST, 2408
- getName, 2406
- INFO, 2408
- INHERIT, 2408
- intValue, 2406
- Level, 2405
- OFF, 2408
- operator<, 2406
- operator==, 2406
- parse, 2406
- SEVERE, 2408
- toString, 2407
- WARNING, 2408
- decaf::util::logging::Logger, 2461
  - ~Logger, 2465
  - addHandler, 2465
  - config, 2465
  - debug, 2466
  - entering, 2466
  - exiting, 2466
  - fine, 2467
  - finer, 2467
  - finest, 2467
  - getAnonymousLogger, 2468
  - getFilter, 2468
  - getHandlers, 2468
  - getLevel, 2468
  - getLogger, 2469
  - getName, 2469
  - getParent, 2469
  - getUseParentHandlers, 2469
  - info, 2470
  - isLoggable, 2470
  - log, 2470, 2471
  - Logger, 2465
  - removeHandler, 2472
  - setFilter, 2472
  - setLevel, 2472
  - setParent, 2472
  - setUseParentHandlers, 2473
  - severe, 2473
  - throwing, 2473
  - warning, 2474
- decaf::util::logging::LoggerHierarchy, 2474
  - ~LoggerHierarchy, 2474
  - LoggerHierarchy, 2474
- decaf::util::logging::LogManager, 2480
  - ~LogManager, 2483
  - addLogger, 2483
  - addPropertyChangeListener, 2484
  - decaf::lang::Runtime, 2487
  - getLogger, 2484
  - getLoggerNames, 2484
  - getLogManager, 2485
  - getProperties, 2485
  - getProperty, 2485
  - LogManager, 2483
  - operator=, 2485
  - readConfiguration, 2485, 2486
  - removePropertyChangeListener, 2486
  - reset, 2486
  - setProperties, 2487
- decaf::util::logging::LogRecord, 2487
  - ~LogRecord, 2489
  - getLevel, 2489
  - getLoggerName, 2489
  - getMessage, 2489
  - getSourceFile, 2489
  - getSourceFunction, 2490
  - getSourceLine, 2490
  - getThrown, 2490
  - getTimestamp, 2490
  - getTreadId, 2490
  - LogRecord, 2489
  - setLevel, 2491
  - setLoggerName, 2491
  - setMessage, 2491
  - setSourceFile, 2491
  - setSourceFunction, 2491
  - setSourceLine, 2492
  - setThrown, 2492
  - setTimestamp, 2492
  - setTreadId, 2492
- decaf::util::logging::LogWriter, 2493
  - ~LogWriter, 2494
  - destroy, 2494
  - getInstance, 2494
  - log, 2494
  - LogWriter, 2494
  - returnInstance, 2494
- decaf::util::logging::MarkBlockLogger, 2563
  - ~MarkBlockLogger, 2564
  - MarkBlockLogger, 2563
- decaf::util::logging::PropertiesChangeListener, 3227
  - ~PropertiesChangeListener, 3227

- onPropertiesReset, 3227
- onPropertyChanged, 3227
- decaf::util::logging::SimpleFormatter, 3604
  - ~SimpleFormatter, 3605
  - format, 3605
  - SimpleFormatter, 3605
- decaf::util::logging::SimpleLogger, 3605
  - ~SimpleLogger, 3606
  - debug, 3606
  - error, 3606
  - fatal, 3606
  - info, 3606
  - log, 3607
  - mark, 3607
  - SimpleLogger, 3606
  - warn, 3607
- decaf::util::logging::StreamHandler, 3756
  - ~StreamHandler, 3758
  - close, 3758
  - flush, 3759
  - isLoggable, 3759
  - publish, 3759
  - setOutputStream, 3759
  - StreamHandler, 3758
- decaf::util::logging::XMLFormatter, 4172
  - ~XMLFormatter, 4173
  - format, 4173
  - getHead, 4173
  - getTail, 4174
  - XMLFormatter, 4173
- decaf::util::Map, 2538
  - ~Map, 2540
  - clear, 2540
  - containsKey, 2541
  - containsValue, 2542
  - copy, 2543
  - equals, 2543
  - get, 2543, 2544
  - isEmpty, 2545
  - keySet, 2546
  - Map, 2540
  - put, 2547
  - putAll, 2547
  - remove, 2548
  - size, 2549
  - values, 2550
- decaf::util::Map::Entry, 1880
  - ~Entry, 1881
  - Entry, 1881
  - getKey, 1881
  - getValue, 1881
  - setValue, 1881
- decaf::util::PriorityQueue, 3116
  - ~PriorityQueue, 3119
  - add, 3120
  - clear, 3120
  - comparator, 3120
  - iterator, 3121
  - offer, 3121
  - operator=, 3122
  - peek, 3122
  - poll, 3122
  - PriorityQueue, 3118, 3119
  - PriorityQueueIterator, 3124
  - remove, 3123
  - size, 3124
- decaf::util::Properties, 3216
  - ~Properties, 3219
  - clear, 3219
  - clone, 3219
  - copy, 3219
  - defaults, 3226
  - equals, 3219
  - getProperty, 3220
  - hasProperty, 3220
  - isEmpty, 3220
  - load, 3221
  - operator=, 3223
  - Properties, 3219
  - propertyNames, 3223
  - remove, 3224
  - setProperty, 3224
  - size, 3224
  - store, 3224, 3225
  - toArray, 3226
  - toString, 3226
- decaf::util::Queue, 3239
  - ~Queue, 3241
  - element, 3241
  - offer, 3241
  - peek, 3242
  - poll, 3242
  - remove, 3243
- decaf::util::Random, 3245
  - next, 3247
  - nextBoolean, 3248
  - nextBytes, 3248, 3249
  - nextDouble, 3249
  - nextFloat, 3249
  - nextGaussian, 3249

- nextInt, 3250
- nextLong, 3250
- Random, 3247
- setSeed, 3251
- decaf::util::Set, 3538
  - ~Set, 3538
- decaf::util::StlList, 3694
  - ~StlList, 3700
  - add, 3700, 3701
  - addAll, 3701
  - clear, 3702
  - contains, 3702
  - copy, 3703
  - equals, 3703
  - get, 3703
  - indexOf, 3703
  - isEmpty, 3704
  - iterator, 3704
  - lastIndexOf, 3704
  - listIterator, 3705, 3706
  - remove, 3706
  - set, 3707
  - size, 3707
  - StlList, 3699, 3700
- decaf::util::StlMap, 3708
  - ~StlMap, 3713
  - clear, 3713
  - containsKey, 3714
  - containsValue, 3714
  - copy, 3714
  - equals, 3715
  - get, 3715, 3716
  - isEmpty, 3716
  - keySet, 3716
  - lock, 3717
  - notify, 3717
  - notifyAll, 3717
  - put, 3718
  - putAll, 3718
  - remove, 3719
  - size, 3719
  - StlMap, 3713
  - tryLock, 3719
  - unlock, 3720
  - values, 3720
  - wait, 3720, 3721
- decaf::util::StlQueue, 3722
  - ~StlQueue, 3725
  - back, 3725
  - clear, 3725
  - empty, 3725
  - enqueueFront, 3725
  - front, 3726
  - getSafeValue, 3726
  - iterator, 3726
  - lock, 3726
  - notify, 3727
  - notifyAll, 3727
  - pop, 3727
  - push, 3728
  - reverse, 3728
  - size, 3728
  - StlQueue, 3725
  - toArray, 3728
  - tryLock, 3728
  - unlock, 3729
  - wait, 3729, 3730
- decaf::util::StlSet, 3731
  - ~StlSet, 3734
  - add, 3734
  - clear, 3735
  - contains, 3735
  - copy, 3736
  - equals, 3736
  - isEmpty, 3736
  - iterator, 3736
  - remove, 3737
  - size, 3737
  - StlSet, 3734
- decaf::util::StringTokenizer, 3779
  - ~StringTokenizer, 3780
  - countTokens, 3780
  - hasMoreTokens, 3780
  - nextToken, 3780, 3781
  - reset, 3781
  - StringTokenizer, 3779
  - toArray, 3781
- decaf::util::Timer, 3901
  - ~Timer, 3904
  - cancel, 3904
  - purge, 3904
  - schedule, 3904–3910
  - scheduleAtFixedRate, 3910–3913
  - Timer, 3904
- decaf::util::TimerTask, 3914
  - ~TimerTask, 3915
  - cancel, 3915
  - decaf::internal::util::TimerTaskHeap, 3916
  - getWhen, 3915

- isScheduled, 3916
- scheduledExecutionTime, 3916
- setScheduledTime, 3916
- Timer, 3916
- TimerImpl, 3916
- TimerTask, 3915
- decaf::util::UUID, 4080
  - ~UUID, 4083
  - clockSequence, 4083
  - compareTo, 4083
  - equals, 4083
  - fromString, 4084
  - getLeastSignificantBits, 4084
  - getMostSignificantBits, 4084
  - nameUUIDFromBytes, 4084, 4085
  - node, 4085
  - operator<, 4085
  - operator==, 4086
  - randomUUID, 4086
  - timestamp, 4086
  - toString, 4087
  - UUID, 4083
  - variant, 4087
  - version, 4087
- decaf::util::zip, 156
- decaf::util::zip::Adler32, 730
  - ~Adler32, 731
  - Adler32, 731
  - getValue, 731
  - reset, 731
  - update, 731, 732
- decaf::util::zip::CheckedInputStream, 1169
  - ~CheckedInputStream, 1171
  - CheckedInputStream, 1170
  - doReadArrayBounded, 1171
  - doReadByte, 1171
  - getChecksum, 1171
  - skip, 1171
- decaf::util::zip::CheckedOutputStream, 1172
  - ~CheckedOutputStream, 1173
  - CheckedOutputStream, 1173
  - doWriteArrayBounded, 1173
  - doWriteByte, 1173
  - getChecksum, 1173
- decaf::util::zip::Checksum, 1174
  - ~Checksum, 1175
  - getValue, 1175
  - reset, 1175
  - update, 1175, 1176
- decaf::util::zip::CRC32, 1568
  - ~CRC32, 1569
  - CRC32, 1569
  - getValue, 1569
  - reset, 1569
  - update, 1569, 1570
- decaf::util::zip::DataFormatException, 1600
  - ~DataFormatException, 1602
  - clone, 1602
  - DataFormatException, 1601, 1602
- decaf::util::zip::Deflater, 1759
  - ~Deflater, 1762
  - BEST\_COMPRESSION, 1768
  - BEST\_SPEED, 1768
  - DEFAULT\_COMPRESSION, 1768
  - DEFAULT\_STRATEGY, 1768
  - deflate, 1762, 1763
  - DEFLATED, 1769
  - Deflater, 1761, 1762
  - end, 1763
  - FILTERED, 1769
  - finish, 1763
  - finished, 1763
  - getAdler, 1764
  - getBytesRead, 1764
  - getBytesWritten, 1764
  - HUFFMAN\_ONLY, 1769
  - needsInput, 1764
  - NO\_COMPRESSION, 1769
  - reset, 1764
  - setDictionary, 1765, 1766
  - setInput, 1766, 1767
  - setLevel, 1767
  - setStrategy, 1768
- decaf::util::zip::DeflaterOutputStream, 1769
  - ~DeflaterOutputStream, 1772
  - buf, 1773
  - close, 1772
  - DEFAULT\_BUFFER\_SIZE, 1773
  - deflate, 1773
  - deflater, 1774
  - DeflaterOutputStream, 1771, 1772
  - doWriteArrayBounded, 1773
  - doWriteByte, 1773
  - finish, 1773
  - isDone, 1774
  - ownDeflater, 1774
- decaf::util::zip::Inflater, 2088
  - ~Inflater, 2090
  - end, 2090
  - finish, 2090

- finished, 2090
- getAdler, 2091
- getBytesRead, 2091
- getBytesWritten, 2091
- getRemaining, 2091
- inflate, 2092, 2093
- Inflater, 2090
- needsDictionary, 2093
- needsInput, 2093
- reset, 2093
- setDictionary, 2094, 2095
- setInput, 2095, 2096
- decaf::util::zip::InflaterInputStream, 2097
  - ~InflaterInputStream, 2101
  - atEOF, 2104
  - available, 2101
  - buff, 2104
  - close, 2102
  - DEFAULT\_BUFFER\_SIZE, 2104
  - doReadArrayBounded, 2102
  - doReadByte, 2102
  - fill, 2102
  - inflator, 2105
  - InflaterInputStream, 2100
  - length, 2105
  - mark, 2102
  - markSupported, 2103
  - ownInflater, 2105
  - reset, 2103
  - skip, 2104
- decaf::util::zip::ZipException, 4175
  - ~ZipException, 4177
  - clone, 4178
  - ZipException, 4176, 4177
- DECAF\_API
  - decaf/util/Config.h, 4275
- DECAF\_CATCH\_EXCEPTION\_CONVERT
  - decaf/lang/exceptions/ExceptionDefines.h, 4241
- DECAF\_CATCH\_NOTHROW
  - decaf/lang/exceptions/ExceptionDefines.h, 4241
- DECAF\_CATCH\_RETHROW
  - decaf/lang/exceptions/ExceptionDefines.h, 4241
- DECAF\_CATCHALL\_NOTHROW
  - decaf/lang/exceptions/ExceptionDefines.h, 4242
- DECAF\_CATCHALL\_THROW
  - decaf/lang/exceptions/ExceptionDefines.h, 4242
- DECAF\_UNUSED
  - decaf/util/Config.h, 4275
- DecafRuntime
  - decaf::internal::DecafRuntime, 1724
- decode
  - decaf::internal::net::URLEncoderDecoder, 4045
  - decaf::lang::Byte, 973
  - decaf::lang::Integer, 2148
  - decaf::lang::Long, 2499
  - decaf::lang::Short, 3542
  - decaf::net::URLDecoder, 4074
- decreaseUsage
  - activemq::util::MemoryUsage, 2594
  - activemq::util::Usage, 4076
- decrementAndGet
  - decaf::util::concurrent::atomic::AtomicInteger, 750
- DedicatedTaskRunner
  - activemq::threads::DedicatedTaskRunner, 1725
- DEF\_MEM\_LEVEL
  - zutil.h, 4639
- DEF\_WBITS
  - zutil.h, 4639
- DEFAULT\_BUFFER\_SIZE
  - decaf::util::zip::DeflaterOutputStream, 1773
  - decaf::util::zip::InflaterInputStream, 2104
- DEFAULT\_COMPRESSION
  - decaf::util::zip::Deflater, 1768
- DEFAULT\_DURABLE\_TOPIC\_PREFETCH
  - activemq::core::policies::DefaultPrefetchPolicy, 1730
- DEFAULT\_MAX\_BLOCK\_SIZE
  - decaf::util::concurrent::ThreadPool, 3894
- DEFAULT\_MAX\_POOL\_SIZE
  - decaf::util::concurrent::ThreadPool, 3894
- DEFAULT\_MESSAGE\_SIZE
  - activemq::commands::Message, 2613
- DEFAULT\_ORDERED\_TARGET
  - activemq::commands::ActiveMQDestination, 322
- DEFAULT\_PRIORITY
  - activemq::cmsutil::CmsTemplate, 1215
- DEFAULT\_QUEUE\_BROWSER\_PREFETCH
  - activemq::core::policies::DefaultPrefetchPolicy, 1730

- DEFAULT\_QUEUE\_PREFETCH
  - activemq::core::policies::DefaultPrefetchPolicy, 4623
  - 1730
- DEFAULT\_STRATEGY
  - decaf::util::zip::Deflater, 1768
- DEFAULT\_TIME\_TO\_LIVE
  - activemq::cmsutil::CmsTemplate, 1215
- DEFAULT\_TOPIC\_PREFETCH
  - activemq::core::policies::DefaultPrefetchPolicy, 4623
  - 1730
- DEFAULT\_URI
  - activemq::core::ActiveMQConnectionFactory, 293
- DEFAULT\_VERSION
  - activemq::wireformat::openwire::OpenWireFormat, 2984
- DefaultPrefetchPolicy
  - activemq::core::policies::DefaultPrefetchPolicy, 1727
- DefaultRedeliveryPolicy
  - activemq::core::policies::DefaultRedeliveryPolicy, 1731
- defaults
  - decaf::util::Properties, 3226
- DefaultServerSocketFactory
  - decaf::internal::net::DefaultServerSocketFactory, 1736
- DefaultSocketFactory
  - decaf::internal::net::DefaultSocketFactory, 1741
- DefaultSSLContext
  - decaf::internal::net::ssl::DefaultSSLContext, 1744
- DefaultSSLServerSocketFactory
  - decaf::internal::net::ssl::DefaultSSLServerSocketFactory, 1747
- DefaultSSLSocketFactory
  - decaf::internal::net::ssl::DefaultSSLSocketFactory, 1753
- deflate
  - Deflater
  - decaf::util::zip::Deflater, 1762, 1763
  - decaf::util::zip::DeflaterOutputStream, deflater, 1773
- deflate.h
  - \_dist\_code, 4623
  - \_length\_code, 4623
  - \_tr\_tally\_dist, 4621
  - \_tr\_tally\_lit, 4622
  - BL\_CODES, 4622
  - BUSY\_STATE, 4623
- Code, 4623
- COMMENT\_STATE, 4623
- ct\_data, 4623
- d\_code, 4623
- D\_CODES, 4623
- Dad, 4623
- deflate\_state, 4623
- EXTRA\_STATE, 4623
- FINISH\_STATE, 4623
- Freq, 4623
- GZIP, 4623
- HCRC\_STATE, 4623
- HEAP\_SIZE, 4623
- INIT\_STATE, 4623
- IsData, 4623
- L\_CODES, 4623
- Len, 4623
- LENGTH\_CODES, 4623
- LITERALS, 4623
- MAX\_BITS, 4623
- MAX\_DIST, 4623
- max\_insert\_length, 4623
- MIN\_LOOKAHEAD, 4623
- NAME\_STATE, 4623
- OF, 4623
- Pos, 4623
- Posf, 4623
- put\_byte, 4623
- static\_tree\_desc, 4623
- tree\_desc, 4623
- WIN\_INIT, 4623
- deflate\_state, 4623
- deflate.h, 4623
- DEFLATED
- decaf::util::zip::Deflater, 1769
- deflateInit
- zlib.h, 4635
- deflateInit2
- zlib.h, 4635
- Deflater
- decaf::util::zip::Deflater, 1761, 1762
- decaf::util::zip::DeflaterOutputStream, 1774
- DeflaterOutputStream
- decaf::util::zip::DeflaterOutputStream, 1771, 1772
- deleteIfCancelled
- decaf::internal::util::TimerTaskHeap, 3918

deliverAcks  
     activemq::core::ActiveMQConsumer, 305  
     activemq::core::ActiveMQSession, 524  
 DELIVERY\_MODE  
     cms::DeliveryMode, 1776  
 deliverySequenceId  
     activemq::commands::MessageDispatchNotification, 2720  
 depth  
     internal\_state, 2191  
 dequeue  
     activemq::core::ActiveMQConsumer, 305  
     activemq::core::MessageDispatchChannel, 2685  
 dequeueNoWait  
     activemq::core::MessageDispatchChannel, 2685  
 deQueueTask  
     decaf::util::concurrent::ThreadPool, 3891  
 descriptor  
     decaf::io::FileDescriptor, 1949  
 destination  
     activemq::cmsutil::CmsTemplate::ProducerExecutor, 3156  
     activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3267  
     activemq::commands::ConsumerControl, 1446  
     activemq::commands::ConsumerInfo, 1509  
     activemq::commands::DestinationInfo, 1784  
     activemq::commands::JournalQueueAck, 2227  
     activemq::commands::JournalTopicAck, 2256  
     activemq::commands::Message, 2613  
     activemq::commands::MessageAck, 2648  
     activemq::commands::MessageDispatch, 2683  
     activemq::commands::MessageDispatchNotification, 2720  
     activemq::commands::MessagePull, 2828  
     activemq::commands::ProducerInfo, 3190  
     activemq::commands::SubscriptionInfo, 3786  
 DESTINATION\_ADD\_OPERATION  
     activemq::core::ActiveMQConstants, 298  
     DESTINATION\_REMOVE\_OPERATION  
     activemq::core::ActiveMQConstants, 298  
     DestinationActions  
     activemq::core::ActiveMQConstants, 298  
     DestinationInfo  
     activemq::commands::DestinationInfo, 1781  
     DestinationInfoMarshaller  
     activemq::wireformat::openwire::marshal::v1::DestinationInfo, 1798  
     activemq::wireformat::openwire::marshal::v2::DestinationInfo, 1786  
     activemq::wireformat::openwire::marshal::v3::DestinationInfo, 1790  
     activemq::wireformat::openwire::marshal::v4::DestinationInfo, 1794  
     activemq::wireformat::openwire::marshal::v5::DestinationInfo, 1807  
     activemq::wireformat::openwire::marshal::v6::DestinationInfo, 1803  
     DestinationOption  
     activemq::core::ActiveMQConstants, 298  
     DestinationType  
     cms::Destination, 1777  
     destOptionMap  
     activemq::core::ActiveMQConstants::StaticInitializer, 3693  
     destOptions  
     activemq::core::ActiveMQConstants::StaticInitializer, 3693  
     destroy  
     activemq::cmsutil::CmsAccessor, 1186  
     activemq::cmsutil::CmsDestinationAccessor, 1189  
     activemq::cmsutil::CmsTemplate, 1205  
     activemq::cmsutil::DestinationResolver, 1811  
     activemq::cmsutil::DynamicDestinationResolver, 1879  
     activemq::cmsutil::ResourceLifecycleManager, 3378  
     activemq::commands::ActiveMQTempQueue, 609  
     activemq::commands::ActiveMQTempTopic, 639



- cms::TemporaryQueue, 3872
- cms::TemporaryTopic, 3873
- decaf::internal::util::concurrent::ConditionImpl, 1292
- decaf::internal::util::concurrent::MutexImpl, 2873
- decaf::util::logging::LogWriter, 2494
- destroyDestination
  - activemq::core::ActiveMQConnection, 268
- destroyMarshallers
  - activemq::wireformat::openwire::OpenWireFormat, 2975
- destroyResources
  - decaf::internal::util::ResourceLifecycleManager, 3376
- DICT
  - inflate.h, 4627
- DICTID
  - inflate.h, 4627
- digit
  - decaf::lang::Character, 1129
- direct
  - gz\_state, 2041
- DISCONNECT
  - activemq::wireformat::stomp::StompCommandConstants, 3740
- DiscoveryEvent
  - activemq::commands::DiscoveryEvent, 1813
- DiscoveryEventMarshaller
  - activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1832
  - activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, 1820
  - activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller, 1824
  - activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller, 1828
  - activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller, 1836
  - activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller, 1816
- dispatch
  - activemq::core::ActiveMQConsumer, 305
  - activemq::core::ActiveMQSession, 525
  - activemq::core::Dispatcher, 1841
- dispatchAsync
- activemq::commands::ConsumerInfo, 1509
- activemq::commands::ProducerInfo, 3190
- activemq::core::DispatchData, 1840
- DIST
  - inflate.h, 4627
- distbits
  - inflate\_state, 2087
- distcode
  - inflate\_state, 2087
- DISTTEXT
  - inflate.h, 4627
- DISTS
  - inftrees.h, 4628
- dl
  - ct\_data\_s, 1571
- dmax
  - inflate\_state, 2087
- doAppendChar
  - decaf::io::Writer, 4136
- doAppendCharSequence
  - decaf::io::Writer, 4137
- doAppendCharSequenceStartEnd
  - decaf::io::Writer, 4137
- doClose
  - activemq::core::ActiveMQConsumer, 305
- doCreateComposite
  - activemq::transport::failover::FailoverTransportFactory, 1944
  - activemq::transport::mock::MockTransportFactory, 2886
  - activemq::transport::tcp::SslTransportFactory, 3085
  - activemq::transport::tcp::TcpTransportFactory, 3079
- doInCms
  - activemq::cmsutil::CmsTemplate::ProducerExecutor, 3156
  - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3266
  - activemq::cmsutil::CmsTemplate::SendExecutor, 3446
- activemq::cmsutil::ProducerCallback, 3154
- activemq::cmsutil::SessionCallback, 3476
- DONE
  - inflate.h, 4627
- done

- gz\_header\_s, 2039
- doReadArray
  - decaf::io::FilterInputStream, 1954
  - decaf::io::InputStream, 2108
  - decaf::io::Reader, 3255
- doReadArrayBounded
  - activemq::io::LoggingInputStream, 2477
  - decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2968
  - decaf::internal::net::tcp::TcpSocketInputStream, 3862
  - decaf::io::BlockingByteArrayInputStream, 847
  - decaf::io::BufferedInputStream, 946
  - decaf::io::ByteArrayInputStream, 1043
  - decaf::io::FilterInputStream, 1954
  - decaf::io::InputStream, 2108
  - decaf::io::InputStreamReader, 2118
  - decaf::io::PushbackInputStream, 3235
  - decaf::io::Reader, 3255
  - decaf::util::zip::CheckedInputStream, 1171
  - decaf::util::zip::InflaterInputStream, 2102
- doReadByte
  - activemq::io::LoggingInputStream, 2475
  - decaf::internal::io::StandardInputStream, 3689
  - decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2968
  - decaf::internal::net::tcp::TcpSocketInputStream, 3862
  - decaf::io::BlockingByteArrayInputStream, 847
  - decaf::io::BufferedInputStream, 946
  - decaf::io::ByteArrayInputStream, 1043
  - decaf::io::FilterInputStream, 1954
  - decaf::io::InputStream, 2108
  - decaf::io::PushbackInputStream, 3235
  - decaf::util::zip::CheckedInputStream, 1171
  - decaf::util::zip::InflaterInputStream, 2102
- doReadChar
  - decaf::io::Reader, 3256
- doReadCharBuffer
  - decaf::io::Reader, 3256
- doReadVector
  - decaf::io::Reader, 3256
- doStartTransaction
  - activemq::core::ActiveMQSession, 525
- Double
  - decaf::lang::Double, 1844
  - DOUBLE\_TYPE
  - activemq::util::PrimitiveValueNode, 3104
  - DoubleArrayBuffer
  - decaf::internal::nio::DoubleArrayBuffer, 1858, 1859
  - DoubleBuffer
  - decaf::io::DoubleBuffer, 1868
  - doubleToLongBits
  - decaf::lang::Double, 1845
  - doubleToRawLongBits
  - decaf::lang::Double, 1846
  - doubleValue
  - activemq::util::PrimitiveValueNode::PrimitiveValue, 3098
  - decaf::lang::Byte, 973
  - decaf::lang::Character, 1130
  - decaf::lang::Double, 1846
  - decaf::lang::Float, 1965
  - decaf::lang::Integer, 2148
  - decaf::lang::Long, 2500
  - decaf::lang::Number, 2919
  - decaf::lang::Short, 3542
  - decaf::util::concurrent::atomic::AtomicInteger, 751
  - doUnmarshal
  - activemq::wireformat::openwire::OpenWireFormat, 2975
  - doWriteArray
  - decaf::io::BufferedOutputStream, 950
  - decaf::io::FilterOutputStream, 1959
  - decaf::io::OutputStream, 2993
  - decaf::io::Writer, 4137
  - doWriteArrayBounded
  - activemq::io::LoggingOutputStream, 2477
  - decaf::internal::io::StandardErrorOutputStream, 3687
  - decaf::internal::io::StandardOutputStream, 3691
  - decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream, 2970
  - decaf::internal::net::tcp::TcpSocketOutputStream, 3865
  - decaf::io::BufferedOutputStream, 950
  - decaf::io::ByteArrayOutputStream, 1048
  - decaf::io::DataOutputStream, 1630
  - decaf::io::FilterOutputStream, 1960
  - decaf::io::OutputStream, 2994
  - decaf::io::OutputStreamWriter, 3001

- decaf::io::Writer, 4137
- decaf::util::zip::CheckedOutputStream, 1173
- decaf::util::zip::DeflaterOutputStream, 1773
- doWriteByte
  - activemq::io::LoggingOutputStream, 2477
  - decaf::internal::io::StandardErrorOutputStream, 3687
  - decaf::internal::io::StandardOutputStream, 3691
  - decaf::internal::net::ssl::openssl::OpenSSLSecureOutputStream, 2971
  - decaf::internal::net::tcp::TcpSocketOutputStream, 3865
  - decaf::io::BufferedOutputStream, 950
  - decaf::io::ByteArrayOutputStream, 1048
  - decaf::io::DataOutputStream, 1630
  - decaf::io::FilterOutputStream, 1960
  - decaf::io::OutputStream, 2994
  - decaf::util::zip::CheckedOutputStream, 1173
  - decaf::util::zip::DeflaterOutputStream, 1773
- doWriteChar
  - decaf::io::Writer, 4137
- doWriteString
  - decaf::io::Writer, 4137
- doWriteStringBounded
  - decaf::io::Writer, 4137
- doWriteVector
  - decaf::io::Writer, 4137
- drainPermits
  - decaf::util::concurrent::Semaphore, 3440
- drainTo
  - decaf::util::concurrent::BlockingQueue, 852
  - decaf::util::concurrent::SynchronousQueue, 3831
- droppable
  - activemq::commands::Message, 2613
- dummy
  - internal\_state, 2191
- duplexConnection
  - activemq::commands::BrokerInfo, 910
- duplicate
  - decaf::internal::nio::ByteArrayBuffer, 1024
- decaf::internal::nio::CharArrayBuffer, 1143
- decaf::internal::nio::DoubleArrayBuffer, 1861
- decaf::internal::nio::FloatArrayBuffer, 1981
- decaf::internal::nio::IntArrayBuffer, 2126
- decaf::internal::nio::LongArrayBuffer, 2518
- decaf::internal::nio::ShortArrayBuffer, 3556
- decaf::nio::ByteBuffer, 1060
- decaf::nio::CharBuffer, 1157
- decaf::nio::DoubleBuffer, 1870
- decaf::nio::FloatBuffer, 1991
- decaf::nio::IntBuffer, 2136
- decaf::nio::LongBuffer, 2528
- decaf::nio::ShortBuffer, 3565
- DUPS\_OK\_ACKNOWLEDGE
  - cms::Session, 3464
- dyn\_dtree
  - internal\_state, 2191
- dyn\_ltree
  - internal\_state, 2191
- dyn\_tree
  - tree\_desc\_s, 4018
- DYN\_TREES
  - zutil.h, 4639
- dynamicCast
  - decaf::lang::Pointer, 3037
- DynamicDestinationResolver
  - activemq::cmsutil::DynamicDestinationResolver, 1879
- E
  - decaf::lang::Math, 2592
- element
  - decaf::util::AbstractQueue, 178
  - decaf::util::Queue, 3241
- empty
  - decaf::util::StlQueue, 3725
- encode
  - decaf::net::URLEncoder, 4075
- encodeOthers
  - decaf::internal::net::URLEncoderDecoder, 4045
- end
- decaf::util::zip::Deflater, 1763
- decaf::util::zip::Inflater, 2090
- ENOUGH

- infrees.h, 4628
- ENOUGH\_DISTS
  - infrees.h, 4628
- ENOUGH\_LENS
  - infrees.h, 4628
- enqueue
  - activemq::core::MessageDispatchChannel, 2686
- enqueueFirst
  - activemq::core::MessageDispatchChannel, 2686
- enqueueFront
  - decaf::util::StlQueue, 3725
- enqueueUsage
  - activemq::util::MemoryUsage, 2594
  - activemq::util::Usage, 4076
- ensureCreated
  - decaf::net::ServerSocket, 3453
  - decaf::net::Socket, 3615
- entering
  - decaf::util::logging::Logger, 2466
- Entry
  - decaf::util::Map::Entry, 1881
- eof
  - gz\_state, 2041
- EOFException
  - decaf::io::EOFException, 1882, 1883
- equals
  - activemq::commands::ActiveMQBlobMessage, 1504
  - 187
  - activemq::commands::ActiveMQBytesMessage, 1538
  - 220
  - activemq::commands::ActiveMQDestination, 1573
  - 316
  - activemq::commands::ActiveMQMapMessage, 1633
  - 354
  - activemq::commands::ActiveMQMessage, 1716
  - 391
  - activemq::commands::ActiveMQMessageTemplate, 1781
  - 423
  - activemq::commands::ActiveMQObjectMessage, 1813
  - 439
  - activemq::commands::ActiveMQQueue, 1896
  - 481
  - activemq::commands::ActiveMQStreamMessage, 2000
  - 540
  - activemq::commands::ActiveMQTempDestination, 161
  - 581
  - activemq::commands::ActiveMQTempQueue, 2226
  - 609
  - activemq::commands::ActiveMQTempTopic, 639
  - activemq::commands::ActiveMQTextMessage, 669
  - activemq::commands::ActiveMQTopic, 699
  - activemq::commands::BaseCommand, 766
  - activemq::commands::BaseDataStructure, 839
  - activemq::commands::BooleanExpression, 862
  - activemq::commands::BrokerId, 876
  - 877
  - activemq::commands::BrokerInfo, 905
  - activemq::commands::ConnectionControl, 1303
  - activemq::commands::ConnectionError, 1334
  - activemq::commands::ConnectionId, 1367
  - activemq::commands::ConnectionInfo, 1395
  - activemq::commands::ConsumerControl, 1443
  - activemq::commands::ConsumerId, 1474
  - 1475
  - activemq::commands::ConsumerInfo, 1504
  - activemq::commands::ControlCommand, 1538
  - activemq::commands::DataArrayResponse, 1573
  - activemq::commands::DataResponse, 1633
  - activemq::commands::DataStructure, 1716
  - activemq::commands::DestinationInfo, 1781
  - activemq::commands::DiscoveryEvent, 1813
  - activemq::commands::ExceptionResponse, 1896
  - activemq::commands::FlushCommand, 2000
  - activemq::commands::IntegerResponse, 2161
  - activemq::commands::JournalQueueAck, 2226
  - activemq::commands::JournalTopicAck, 2226

- 2254
- activemq::commands::JournalTrace, 2282
- activemq::commands::JournalTransaction, 2309
- activemq::commands::KeepAliveInfo, 2337
- activemq::commands::LastPartialCommand, 2372
- activemq::commands::LocalTransactionId, 2422, 2423
- activemq::commands::Message, 2602
- activemq::commands::MessageAck, 2645
- activemq::commands::MessageDispatch, 2680
- activemq::commands::MessageDispatchNotification, 2718
- activemq::commands::MessageId, 2753
- activemq::commands::MessagePull, 2826
- activemq::commands::NetworkBridgeFilter, 2879
- activemq::commands::PartialCommand, 3003
- activemq::commands::ProducerAck, 3126
- activemq::commands::ProducerId, 3159
- activemq::commands::ProducerInfo, 3187
- activemq::commands::RemoveInfo, 3286
- activemq::commands::RemoveSubscriptionInfo, 3315
- activemq::commands::ReplayCommand, 3345
- activemq::commands::Response, 3381
- activemq::commands::SessionId, 3478, 3479
- activemq::commands::SessionInfo, 3509
- activemq::commands::ShutdownInfo, 3574
- activemq::commands::SubscriptionInfo, 3784
- activemq::commands::TransactionId, 3934
- activemq::commands::TransactionInfo, 3961
- activemq::commands::WireFormatInfo, 4097
- activemq::commands::XATransactionId, 4145
- decaf::lang::Boolean, 858
- decaf::lang::Byte, 973, 974
- decaf::lang::Character, 1130
- decaf::lang::Comparable, 1249
- decaf::lang::Double, 1846, 1847
- decaf::lang::Float, 1966
- decaf::lang::Integer, 2149
- decaf::lang::Long, 2500
- decaf::lang::Short, 3543
- decaf::net::URI, 4037
- decaf::nio::ByteBuffer, 1061
- decaf::nio::CharBuffer, 1157
- decaf::nio::DoubleBuffer, 1871
- decaf::nio::FloatBuffer, 1991
- decaf::nio::IntBuffer, 2136
- decaf::nio::LongBuffer, 2528
- decaf::nio::ShortBuffer, 3566
- decaf::security::cert::Certificate, 1113
- decaf::security::Principal, 3115
- decaf::util::AbstractCollection, 166
- decaf::util::Collection, 1222
- decaf::util::concurrent::ConcurrentStlMap, 1272
- decaf::util::concurrent::SynchronousQueue, 3832
- decaf::util::concurrent::TimeUnit, 3923
- decaf::util::Date, 1721
- decaf::util::logging::Level, 2406
- decaf::util::Map, 2543
- decaf::util::Properties, 3219
- decaf::util::StlList, 3703
- decaf::util::StlMap, 3715
- decaf::util::StlSet, 3736
- decaf::util::UUID, 4083
- decaf::io::FileDescriptor, 1949
- gz\_state, 2041
- ERR\_MSG
- util.h, 4639
- ERR\_RETURN
- util.h, 4639
- Error
- decaf::util::logging, 156
- error
- decaf::util::logging::ErrorManager, 1885
- decaf::util::logging::SimpleLogger, 3606
- ERROR\_CMD
- activemq::wireformat::stomp::StompCommandConstants, 3740
- ErrorManager
- decaf::util::logging::ErrorManager, 1885
- Exception
- decaf::lang::Exception, 1888, 1889
- exception

- activemq::commands::ConnectionError, inflate\_state, 2087
- 1336 extra\_len
- activemq::commands::ExceptionResponse, gz\_header\_s, 2039
- 1897 extra\_max
- ExceptionResponse gz\_header\_s, 2039
- activemq::commands::ExceptionResponse, EXTRA\_STATE
- 1896 deflate.h, 4623
- ExceptionResponseMarshaller
- activemq::wireformat::openwire::marshaller::v1::ExceptionResponseMarshaller, EL\_OPEN
- 1920 zutil.h, 4639
- activemq::wireformat::openwire::marshaller::v2::ExceptionResponseMarshaller, failIfReadOnlyBody
- 1903 activemq::commands::ActiveMQMessageTemplate,
- activemq::wireformat::openwire::marshaller::v3::ExceptionResponseMarshaller, failIfReadOnlyProperties
- 1907
- activemq::wireformat::openwire::marshaller::v4::ExceptionResponseMarshaller, failIfReadOnlyBody
- 1916 423
- activemq::wireformat::openwire::marshaller::v5::ExceptionResponseMarshaller, failIfReadOnlyBody
- 1912 activemq::commands::ActiveMQMessageTemplate,
- activemq::wireformat::openwire::marshaller::v6::ExceptionResponseMarshaller, failIfReadOnlyBody
- 1899 423
- FailoverTransport
- exclusive activemq::transport::failover::FailoverTransport,
- activemq::commands::ActiveMQDestination, 1933
- 322 FailoverTransportListener
- activemq::commands::ConsumerInfo, activemq::transport::failover::FailoverTransport,
- 1509 1942
- execute activemq::transport::failover::FailoverTransportListener,
- activemq::cmsutil::CmsTemplate, 1205, 1946
- 1206 FAR
- activemq::core::ActiveMQSessionExecutor, zconf.h, 4632
- 534 Fatal
- decaf::util::concurrent::Executor, 1928 decaf::util::logging, 156
- executeFirst fatal
- activemq::core::ActiveMQSessionExecutor, decaf::util::logging::SimpleLogger, 3606
- 534 faultTolerant
- ExecutionException activemq::commands::ConnectionControl,
- decaf::util::concurrent::ExecutionException, 1306
- 1924, 1925 activemq::commands::ConnectionInfo,
- exit 1399
- activemq::commands::ConnectionControl, faultTolerantConfiguration
- 1306 activemq::commands::BrokerInfo, 910
- exiting fc
- decaf::util::logging::Logger, 2466 ct\_data\_s, 1571
- EXLEN fd
- inflate.h, 4626 decaf::net::SocketImpl, 3644
- expiration gz\_state, 2041
- activemq::commands::Message, 2613 FileDescriptor
- EXTRA decaf::io::FileDescriptor, 1948
- inflate.h, 4627 FileName
- extra activemq::commands::BrokerError::StackTraceElement,
- gz\_header\_s, 2039 3686

- fill
  - decaf::util::zip::InflaterInputStream, 2102
- FILTERED
  - decaf::util::zip::Deflater, 1769
- FilterInputStream
  - decaf::io::FilterInputStream, 1953
- FilterOutputStream
  - decaf::io::FilterOutputStream, 1959
- find
  - decaf::internal::util::TimerTaskHeap, 3918
- findFactory
  - activemq::transport::TransportRegistry, 4016
  - activemq::wireformat::WireFormatRegistry, 4131
- FINE
  - decaf::util::logging::Level, 2407
- fine
  - decaf::util::logging::Logger, 2467
- FINER
  - decaf::util::logging::Level, 2408
- finer
  - decaf::util::logging::Logger, 2467
- FINEST
  - decaf::util::logging::Level, 2408
- finest
  - decaf::util::logging::Logger, 2467
- finish
  - decaf::util::zip::Deflater, 1763
  - decaf::util::zip::DeflaterOutputStream, 1773
  - decaf::util::zip::Inflater, 2090
- FINISH\_STATE
  - deflate.h, 4623
- finished
  - decaf::util::zip::Deflater, 1763
  - decaf::util::zip::Inflater, 2090
- fire
  - activemq::core::ActiveMQConnection, 269
  - activemq::core::ActiveMQSession, 525
  - activemq::transport::TransportFilter, 4007
- fireCommand
  - activemq::transport::mock::MockTransport, 2857
- fireException
  - activemq::transport::mock::MockTransport, 2857
- firstMessageId
  - activemq::commands::MessageAck, 2648
  - activemq::commands::ReplayCommand, 3347
- FLAGS
  - inflate.h, 4626
- flags
  - inflate\_state, 2087
- flip
  - decaf::nio::Buffer, 939
- Float
  - decaf::lang::Float, 1964
  - FloatArrayBuffer
    - decaf::internal::nio::FloatArrayBuffer, 1978, 1979
- FloatBuffer
  - decaf::nio::FloatBuffer, 1988
- floatToIntBits
  - decaf::lang::Float, 1966
- floatToRawIntBits
  - decaf::lang::Float, 1967
- floatValue
  - activemq::util::PrimitiveValueNode::PrimitiveValue, 3098
  - decaf::lang::Byte, 974
  - decaf::lang::Character, 1130
  - decaf::lang::Double, 1847
  - decaf::lang::Float, 1967
  - decaf::lang::Integer, 2149
  - decaf::lang::Long, 2501
  - decaf::lang::Number, 2919
  - decaf::lang::Short, 3543
  - decaf::util::concurrent::atomic::AtomicInteger, 751
- floor
  - decaf::lang::Math, 2580
- flush
  - activemq::commands::ConsumerControl, 1446
  - decaf::internal::io::StandardErrorOutputStream, 3688
  - decaf::internal::io::StandardOutputStream, 3691
  - decaf::io::BufferedOutputStream, 950
  - decaf::io::FilterOutputStream, 1960
  - decaf::io::Flushable, 1998
  - decaf::io::OutputStream, 2994
  - decaf::io::OutputStreamWriter, 3001

Generated on Tue Mar 8 2011 17:10:05 for activemq-cpp-3.2.5 by Doxygen



- getAckMode
  - activemq::commands::SessionInfo, 3507
- getAcknowledgeMode
  - activemq::cmsutil::PooledSession, 3053
- getBackup
  - activemq::core::ActiveMQSession, 525
  - cms::Session, 3473
- getBackupPoolSize
  - activemq::transport::failover::BackupTransportPool, 763
- getAckType
  - activemq::commands::MessageAck, 2645
- getAdditionalPredicate
  - activemq::commands::ConsumerInfo, 1504, 1505
- getAddress
  - decaf::net::InetAddress, 2080
- getAdler
  - decaf::util::zip::Deflater, 1764
  - decaf::util::zip::Inflater, 2091
- getAlgorithm
  - decaf::security::Key, 2365
- getAndAdd
  - decaf::util::concurrent::atomic::AtomicInteger, 751
- getAndDecrement
  - decaf::util::concurrent::atomic::AtomicInteger, 752
- getAndIncrement
  - decaf::util::concurrent::atomic::AtomicInteger, 752
- getAndSet
  - decaf::util::concurrent::atomic::AtomicBoolean, 747
  - decaf::util::concurrent::atomic::AtomicInteger, 752
  - decaf::util::concurrent::atomic::AtomicReference, 758
- getAnonymousLogger
  - decaf::util::logging::Logger, 2468
- getAnyAddress
  - decaf::net::InetAddress, 2080
- getAprPool
  - decaf::internal::AprPool, 736
- getArrival
  - activemq::commands::Message, 2603
- getAuthority
  - decaf::internal::net::URIType, 4066
  - decaf::net::URI, 4037
- getBacklog
  - decaf::util::concurrent::ThreadPool, 3891
- getBackOffMultiplier
  - activemq::core::policies::DefaultRedeliveryPolicy, 1731
- activemq::core::RedeliveryPolicy, 3269
- activemq::transport::failover::FailoverTransport, 1934
- getBackup
  - activemq::transport::failover::BackupTransportPool, 763
- getBackupPoolSize
  - activemq::transport::failover::BackupTransportPool, 763
- getBasicConstraints
  - decaf::security::cert::X509Certificate, 4142
- getBlockSize
  - decaf::util::concurrent::ThreadPool, 3891
- getBody
  - activemq::wireformat::stomp::StompFrame, 3743
- getBodyBytes
  - activemq::commands::ActiveMQBytesMessage, 220
- getBodyLength
  - activemq::commands::ActiveMQBytesMessage, 220
- can
  - activemq::wireformat::stomp::StompFrame, 3743
- can
  - activemq::commands::ActiveMQBytesMessage, 1084
- getBool
  - activemq::util::PrimitiveList, 3070
- getBoolean
  - activemq::util::PrimitiveMap, 3082
  - activemq::util::PrimitiveValueNode, 3107
- activemq::commands::ActiveMQMapMessage, 355
- cms::MapMessage, 2554
- getBooleanProperty
  - activemq::commands::ActiveMQMessageTemplate, 423
- activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2820
- cms::Message, 2620
- getBranchQualifier
  - activemq::commands::XATransactionId, 4145
- getBrokerId
  - activemq::commands::BrokerInfo, 905,
- getBrokerInTime

activemq::commands::Message, 2603  
 getBrokerName  
     activemq::commands::BrokerInfo, 906  
     activemq::commands::DiscoveryEvent, getBytesRead  
         1814  
 getBrokerOutTime  
     activemq::commands::Message, 2603  
 getBrokerPath  
     activemq::commands::ConnectionInfo, 1395, 1396  
     activemq::commands::ConsumerInfo, 1505  
     activemq::commands::DestinationInfo, 1782  
     activemq::commands::Message, 2603  
     activemq::commands::ProducerInfo, 3187, 3188  
 getBrokerSequenceId  
     activemq::commands::MessageId, 2753  
 getBrokerUploadUrl  
     activemq::commands::BrokerInfo, 906  
 getBrokerURL  
     activemq::commands::BrokerInfo, 906  
     activemq::core::ActiveMQConnection, 269  
     activemq::core::ActiveMQConnectionFactory, 286  
 getByAddress  
     decaf::net::InetAddress, 2080, 2081  
 getByte  
     activemq::commands::ActiveMQMapMessage, 355  
     activemq::util::PrimitiveList, 3071  
     activemq::util::PrimitiveMap, 3083  
     activemq::util::PrimitiveValueNode, 3107  
     cms::MapMessage, 2554  
 getByteArray  
     activemq::util::PrimitiveList, 3071  
     activemq::util::PrimitiveMap, 3083  
     activemq::util::PrimitiveValueNode, 3108  
     decaf::internal::util::ByteArrayAdapter, 988  
 getByteProperty  
     activemq::commands::ActiveMQMessageTemplate, 424  
     activemq::wireformat::openwire::utils::MessagePropertyIntrospector, 2820  
     cms::Message, 2621  
 getBytes  
     activemq::commands::ActiveMQMapMessage, 356  
     cms::MapMessage, 2555  
     decaf::util::zip::Deflater, 1764  
     decaf::util::zip::Inflater, 2091  
     getBytesWritten  
         decaf::util::zip::Deflater, 1764  
         decaf::util::zip::Inflater, 2091  
     getCacheSize  
         activemq::commands::WireFormatInfo, 4097  
     activemq::wireformat::openwire::OpenWireFormat, 2976  
     getCapacity  
         decaf::internal::util::ByteArrayAdapter, 988  
     getCause  
         activemq::commands::BrokerError, 870  
         cms::CMSEException, 1192  
         decaf::lang::Exception, 1890  
         decaf::lang::Throwable, 3897  
     getChar  
         activemq::commands::ActiveMQMapMessage, 356  
         activemq::util::PrimitiveList, 3072  
         activemq::util::PrimitiveMap, 3084  
         activemq::util::PrimitiveValueNode, 3108  
         cms::MapMessage, 2555  
         decaf::internal::nio::ByteBuffer, 1025  
         decaf::internal::util::ByteArrayAdapter, 988  
         decaf::nio::ByteBuffer, 1063  
     getCharArray  
         decaf::internal::util::ByteArrayAdapter, 989  
     getCharCapacity  
         decaf::internal::util::ByteArrayAdapter, 989  
     getChecksum  
         decaf::util::zip::CheckedInputStream, 1171  
         decaf::util::zip::CheckedOutputStream, 1171  
     getCipherSuites  
         decaf::internal::util::ByteArrayAdapter, 989  
     getPropertiesIntrospector, 3660  
     getClientID  
         activemq::core::ActiveMQConnection, 269

---

cms::Connection, 1299  
 getClientId  
 activemq::commands::ActiveMQDestination, 317  
 activemq::commands::ConnectionInfo, 1396  
 activemq::commands::JournalTopicAck, 2254  
 activemq::commands::RemoveSubscriptionInfo, 3315, 3316  
 activemq::commands::SubscriptionInfo, 3784  
 activemq::core::ActiveMQConnectionFactory, 287  
 getCloseTimeout  
 activemq::core::ActiveMQConnection, 270  
 activemq::core::ActiveMQConnectionFactory, 287  
 getCluster  
 activemq::commands::Message, 2603  
 getCMSCorrelationID  
 activemq::commands::ActiveMQMessageTemplate, 424  
 cms::Message, 2622  
 getCMSDeliveryMode  
 activemq::commands::ActiveMQMessageTemplate, 424  
 cms::Message, 2622  
 getCMSDestination  
 activemq::commands::ActiveMQDestination, 317  
 activemq::commands::ActiveMQMessageTemplate, 425  
 activemq::commands::ActiveMQQueue, 482  
 activemq::commands::ActiveMQTempQueue, 609  
 activemq::commands::ActiveMQTempTopic, 639  
 activemq::commands::ActiveMQTopic, 699  
 cms::Message, 2623  
 getCMSExpiration  
 activemq::commands::ActiveMQMessageTemplate, 425  
 cms::Message, 2623  
 getCMSMajorVersion  
 activemq::core::ActiveMQConnectionMetaData, 294  
 cms::ConnectionMetaData, 1426  
 getCMSMessageID  
 activemq::commands::ActiveMQMessageTemplate, 425  
 cms::Message, 2624  
 getCMSMinorVersion  
 activemq::core::ActiveMQConnectionMetaData, 294  
 cms::ConnectionMetaData, 1427  
 getCMSPriority  
 activemq::commands::ActiveMQMessageTemplate, 426  
 cms::Message, 2625  
 getCMSProperties  
 activemq::commands::ActiveMQQueue, 482  
 activemq::commands::ActiveMQTempQueue, 610  
 activemq::commands::ActiveMQTempTopic, 639  
 activemq::commands::ActiveMQTopic, 700  
 cms::Destination, 1778  
 getCMSProviderName  
 activemq::core::ActiveMQConnectionMetaData, 295  
 cms::ConnectionMetaData, 1427  
 getCMSRedelivered  
 activemq::commands::ActiveMQMessageTemplate, 426  
 cms::Message, 2625  
 getCMSReplyTo  
 activemq::commands::ActiveMQMessageTemplate, 426  
 cms::Message, 2626  
 getCMSTimestamp  
 activemq::commands::ActiveMQMessageTemplate, 427  
 cms::Message, 2626  
 getCMSType  
 activemq::commands::ActiveMQMessageTemplate, 427  
 cms::Message, 2627  
 getCMSVersion  
 activemq::core::ActiveMQConnectionMetaData, 295  
 cms::ConnectionMetaData, 1427  
 getCMSXPropertyNames  
 activemq::core::ActiveMQConnectionMetaData, 295

---

- cms::ConnectionMetaData, 1428
- getCollisionAvoidancePercent
  - activemq::core::policies::DefaultRedeliveryPolicy, 1443, 1444
  - 1732
  - activemq::core::RedeliveryPolicy, 3270
- getCommand
  - activemq::commands::ControlCommand, 1538
  - activemq::wireformat::stomp::StompFrame, 3744
- getCommandId
  - activemq::commands::BaseCommand, 767
  - activemq::commands::Command, 1228
  - activemq::commands::PartialCommand, 3004
- getCommands
  - activemq::state::TransactionState, 3991
- getComponents
  - activemq::util::CompositeData, 1254
- getConnectedBrokers
  - activemq::commands::ConnectionControl, 1303, 1304
- getConnection
  - activemq::commands::Message, 2603
  - activemq::core::ActiveMQSession, 525
- getConnectionFactory
  - activemq::cmsutil::CmsAccessor, 1186
- getConnectionId
  - activemq::commands::BrokerInfo, 906
  - activemq::commands::ConnectionError, 1334
  - activemq::commands::ConnectionInfo, 1396
  - activemq::commands::ConsumerId, 1475
  - activemq::commands::DestinationInfo, 1782
  - activemq::commands::LocalTransactionId, 2423
  - activemq::commands::ProducerId, 3159
  - activemq::commands::RemoveSubscriptionInfo, 3316
  - activemq::commands::SessionId, 3479
  - activemq::commands::TransactionInfo, 3961, 3962
  - activemq::core::ActiveMQConnection, 270
- getConnectionInfo
  - activemq::core::ActiveMQConnection, 270
- getConsumerId
  - activemq::commands::ConsumerControl, 1443, 1444
  - activemq::commands::ConsumerInfo, 1505
  - activemq::commands::MessageAck, 2645
  - activemq::commands::MessageDispatch, 2681
  - activemq::commands::MessageDispatchNotification, 2718
  - activemq::commands::MessagePull, 2826
  - activemq::core::ActiveMQConsumer, 306
  - activemq::core::DispatchData, 1840
  - activemq::core::ActiveMQConsumer, 306
- getConsumerState
  - activemq::state::SessionState, 3537
- getConsumerStates
  - activemq::state::SessionState, 3537
- getContent
  - activemq::commands::Message, 2603, 2604
- getContext
  - decaf::internal::net::ssl::DefaultSSLContext, 1744
- getCorrelationId
  - activemq::commands::Message, 2604
  - activemq::commands::MessagePull, 2826
  - activemq::commands::Response, 3381
- getCount
  - decaf::util::concurrent::CountDownLatch, 1568
- getData
  - activemq::commands::DataArrayResponse, 1573, 1574
  - activemq::commands::DataResponse, 1633, 1634
  - activemq::commands::PartialCommand, 3004
- getDataStructure
  - activemq::commands::Message, 2604
- getDataStructureType
  - activemq::commands::ActiveMQBlobMessage, 187
  - activemq::commands::ActiveMQBytesMessage, 221
  - activemq::commands::ActiveMQDestination, 317

activemq::commands::ActiveMQMapMessage,	activemq::commands::FlushCommand,
356	2000
activemq::commands::ActiveMQMessage,	activemq::commands::IntegerResponse,
392	2161
activemq::commands::ActiveMQObjectMessage,	activemq::commands::JournalQueueAck,
440	2226
activemq::commands::ActiveMQQueue,	activemq::commands::JournalTopicAck,
482	2254
activemq::commands::ActiveMQStreamMessage,	activemq::commands::JournalTrace,
540	2282
activemq::commands::ActiveMQTempDestination,	activemq::commands::JournalTransaction,
581	2310
activemq::commands::ActiveMQTempQueue,	activemq::commands::KeepAliveInfo,
610	2337
activemq::commands::ActiveMQTempTopic,	activemq::commands::LastPartialCommand,
640	2373
activemq::commands::ActiveMQTextMessage,	activemq::commands::LocalTransactionId,
669	2423
activemq::commands::ActiveMQTopic,	activemq::commands::Message,
700	2604
activemq::commands::BrokerError,	activemq::commands::MessageAck,
871	2645
activemq::commands::BrokerId,	activemq::commands::MessageDispatch,
877	2681
activemq::commands::BrokerInfo,	activemq::commands::MessageDispatchNotification,
906	2718
activemq::commands::ConnectionControl,	activemq::commands::MessageId,
1304	2753
activemq::commands::ConnectionError,	activemq::commands::MessagePull,
1334	2826
activemq::commands::ConnectionId,	activemq::commands::NetworkBridgeFilter,
1367	2879
activemq::commands::ConnectionInfo,	activemq::commands::PartialCommand,
1396	3004
activemq::commands::ConsumerControl,	activemq::commands::ProducerAck,
1444	3126
activemq::commands::ConsumerId,	activemq::commands::ProducerId,
1475	3159
activemq::commands::ConsumerInfo,	activemq::commands::ProducerInfo,
1505	3188
activemq::commands::ControlCommand,	activemq::commands::RemoveInfo,
1538	3286
activemq::commands::DataArrayResponse,	activemq::commands::RemoveSubscriptionInfo,
1574	3316
activemq::commands::DataResponse,	activemq::commands::ReplayCommand,
1634	3345
activemq::commands::DataStructure,	activemq::commands::Response,
1717	3381
activemq::commands::DestinationInfo,	activemq::commands::SessionId,
1782	3479
activemq::commands::DiscoveryEvent,	activemq::commands::SessionInfo,
1814	3507
activemq::commands::ExceptionResponse,	activemq::commands::ShutdownInfo,
1896	3574
	activemq::commands::SubscriptionInfo,
	3784
	activemq::commands::TransactionId,
	3934
	activemq::commands::TransactionInfo,
	3962
	activemq::commands::WireFormatInfo,
	4098

activemq::commands::XATransactionId,      activemq::wireformat::openwire::marshal::v1::DestinationInfo  
 4145      1799  
 activemq::wireformat::openwire::marshal::DataStreamMarshaller,      activemq::wireformat::openwire::marshal::v1::DiscoveryEvent  
 1668      1833  
 activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller,      activemq::wireformat::openwire::marshal::v1::ExceptionResponse  
 195      1920  
 activemq::wireformat::openwire::marshal::v1::ActiveMQBufferMessageMarshaller,      activemq::wireformat::openwire::marshal::v1::FlushCommand  
 240      2020  
 activemq::wireformat::openwire::marshal::v1::ActiveMQDataMessageMarshaller,      activemq::wireformat::openwire::marshal::v1::IntegerResponse  
 370      2181  
 activemq::wireformat::openwire::marshal::v1::ActiveMQErrorMessageMarshaller,      activemq::wireformat::openwire::marshal::v1::JournalQueueAck  
 398      2249  
 activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller,      activemq::wireformat::openwire::marshal::v1::JournalTopicAck  
 446      2278  
 activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller,      activemq::wireformat::openwire::marshal::v1::JournalTraceMessage  
 492      2301  
 activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller,      activemq::wireformat::openwire::marshal::v1::JournalTransaction  
 559      2333  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTopicQueueMarshaller,      activemq::wireformat::openwire::marshal::v1::KeepAliveInfo  
 617      2361  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller,      activemq::wireformat::openwire::marshal::v1::LastPartialCommand  
 651      2396  
 activemq::wireformat::openwire::marshal::v1::ActiveMQBufferMessageMarshaller,      activemq::wireformat::openwire::marshal::v1::LocalTransaction  
 681      2447  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller,      activemq::wireformat::openwire::marshal::v1::MessageAcknowledge  
 711      2667  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireMarshaller,      activemq::wireformat::openwire::marshal::v1::MessageDispatch  
 888      2709  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireMarshaller,      activemq::wireformat::openwire::marshal::v1::MessageDispatch  
 920      2739  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireMarshaller,      activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller  
 1317      2777  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireMarshaller,      activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller  
 1350      2847  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireMarshaller,      activemq::wireformat::openwire::marshal::v1::NetworkBridge  
 1382      2902  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireMarshaller,      activemq::wireformat::openwire::marshal::v1::PartialCommand  
 1414      3029  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireMarshaller,      activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller  
 1461      3151  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireMarshaller,      activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller  
 1491      3183  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireMarshaller,      activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller  
 1524      3200  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireMarshaller,      activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller  
 1554      3302  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireMarshaller,      activemq::wireformat::openwire::marshal::v1::RemoveSubscriber  
 1589      3319  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireMarshaller,      activemq::wireformat::openwire::marshal::v1::ReplayCommand  
 1657      3353

activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller, 3410  
 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller, 3502  
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller, 3518  
 activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller, 3586  
 activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller, 3792  
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller, 3970  
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller, 4122  
 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, 4161  
 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller, 204  
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller, 257  
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller, 383  
 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller, 411  
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller, 459  
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller, 505  
 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller, 571  
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller, 629  
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller, 659  
 activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller, 694  
 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller, 724  
 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller, 900  
 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller, 933  
 activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller, 1329  
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller, 1337  
 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller, 1370  
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller, 1401

activemq::wireformat::openwire::marshal::v2::ProduceAckMarshalleropenwire::marshal::v3::BrokerIdMarsh  
 3130 879  
 activemq::wireformat::openwire::marshal::v2::ProduceWithMarshalleropenwire::marshal::v3::BrokerInfoMar  
 3162 912  
 activemq::wireformat::openwire::marshal::v2::ProduceWithMarshalleropenwire::marshal::v3::ConnectionCon  
 3196 1308  
 activemq::wireformat::openwire::marshal::v2::RemoveWireMarshalleropenwire::marshal::v3::ConnectionErro  
 3289 1342  
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshalleropenwire::marshal::v3::ConnectionIdM  
 3328 1374  
 activemq::wireformat::openwire::marshal::v2::ReplyCorrelationMarshalleropenwire::marshal::v3::ConnectionInfo  
 3357 1405  
 activemq::wireformat::openwire::marshal::v2::ResponseMarshalleropenwire::marshal::v3::ConsumerCont  
 3395 1452  
 activemq::wireformat::openwire::marshal::v2::SessionIdMarshalleropenwire::marshal::v3::ConsumerIdMa  
 3482 1482  
 activemq::wireformat::openwire::marshal::v2::SessionWireMarshalleropenwire::marshal::v3::ConsumerInfoM  
 3527 1515  
 activemq::wireformat::openwire::marshal::v2::StartupWireMarshalleropenwire::marshal::v3::ControlComma  
 3582 1545  
 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshalleropenwire::marshal::v3::DataArrayResp  
 3808 1580  
 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshalleropenwire::marshal::v3::DataResponseM  
 3987 1649  
 activemq::wireformat::openwire::marshal::v2::WireFormatIdMarshalleropenwire::marshal::v3::DestinationInfo  
 4114 1790  
 activemq::wireformat::openwire::marshal::v2::XidTransactionIdMarshalleropenwire::marshal::v3::DiscoveryEven  
 4153 1825  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBrokerMessageMarshalleropenwire::marshal::v3::ExceptionResp  
 191 1908  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBrokerMessageMarshalleropenwire::marshal::v3::FlushCommand  
 236 2012  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBrokerMessageMarshalleropenwire::marshal::v3::IntegerRespons  
 366 2172  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBrokerMessageMarshalleropenwire::marshal::v3::JournalQueueA  
 394 2241  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBrokerMessageMarshalleropenwire::marshal::v3::JournalTopicAc  
 442 2266  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBrokerMessageMarshalleropenwire::marshal::v3::JournalTraceM  
 488 2289  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBrokerMessageMarshalleropenwire::marshal::v3::JournalTransac  
 554 2321  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBrokerMessageMarshalleropenwire::marshal::v3::KeepAliveInfo  
 612 2348  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBrokerMessageMarshalleropenwire::marshal::v3::LastPartialCom  
 642 2379  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBrokerMessageMarshalleropenwire::marshal::v3::LocalTransacti  
 673 2435  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBrokerMessageMarshalleropenwire::marshal::v3::MessageAckM  
 703 2658



activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	2696
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	497
activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller	2730
activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller	563
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	2769
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	621
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	2838
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	647
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	2894
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	677
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	3020
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	707
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	3138
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	883
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	3171
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	916
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	3209
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	1312
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	3298
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	1346
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	3324
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	1378
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	3361
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	1410
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller	3405
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller	1457
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	3498
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	1487
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	3523
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	1520
activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller	3594
activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller	1549
activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller	3788
activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller	1584
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller	3974
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller	1653
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller	4126
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller	1794
activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller	4165
activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller	1829
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller	199
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller	1916
activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller	245
activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller	2016
activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller	374
activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller	2177
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	403
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	2245
activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller	450
activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller	2274

activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller,marshal::v4::XATransactionIdMarshaller 4157  
 2297  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMarshaller,marshal::v5::ActiveMQBlobMarshaller 208  
 2329  
 activemq::wireformat::openwire::marshal::v4::ActiveMQByteMarshaller,marshal::v5::ActiveMQByteMarshaller 249  
 2352  
 activemq::wireformat::openwire::marshal::v4::ActiveMQMapMarshaller,marshal::v5::ActiveMQMapMarshaller 379  
 2392  
 activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller,marshal::v5::ActiveMQMessageMarshaller 407  
 2443  
 activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMarshaller,marshal::v5::ActiveMQObjectMarshaller 455  
 2663  
 activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller,marshal::v5::ActiveMQQueueMarshaller 501  
 2705  
 activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMarshaller,marshal::v5::ActiveMQStreamMarshaller 567  
 2735  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempObjectMarshaller,marshal::v5::ActiveMQTempObjectMarshaller 625  
 2761  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempObjectMarshaller,marshal::v5::ActiveMQTempObjectMarshaller 655  
 2843  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTextMarshaller,marshal::v5::ActiveMQTextMarshaller 686  
 2898  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller,marshal::v5::ActiveMQTopicMarshaller 715  
 3025  
 activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller,marshal::v5::BrokerIdMarshaller 892  
 3134  
 activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller,marshal::v5::BrokerInfoMarshaller 925  
 3167  
 activemq::wireformat::openwire::marshal::v4::ConnectionContainerMarshaller,marshal::v5::ConnectionContainerMarshaller 1321  
 3192  
 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller,marshal::v5::ConnectionErrorMarshaller 1354  
 3310  
 activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller,marshal::v5::ConnectionIdMarshaller 1386  
 3341  
 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller,marshal::v5::ConnectionInfoMarshaller 1418  
 3348  
 activemq::wireformat::openwire::marshal::v4::ConsumerContainerMarshaller,marshal::v5::ConsumerContainerMarshaller 1465  
 3390  
 activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller,marshal::v5::ConsumerIdMarshaller 1495  
 3486  
 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller,marshal::v5::ConsumerInfoMarshaller 1528  
 3531  
 activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller,marshal::v5::ControlCommandMarshaller 1558  
 3599  
 activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller,marshal::v5::DataArrayResponseMarshaller 1593  
 3800  
 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller,marshal::v5::DataResponseMarshaller 1636  
 3982  
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller,marshal::v5::DestinationInfoMarshaller 1807  
 4118

activemq::wireformat::openwire::marshal::v5::DiscoveryInfoMarshaller,	1837
activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller,	1912
activemq::wireformat::openwire::marshal::v5::FlushQueueInfoMarshaller,	2024
activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller,	2185
activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller,	2237
activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller,	2258
activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller,	2305
activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller,	2325
activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller,	2357
activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller,	2388
activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller,	2439
activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller,	2671
activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller,	2700
activemq::wireformat::openwire::marshal::v5::MessageDispatchNoOperationMarshaller,	2743
activemq::wireformat::openwire::marshal::v5::MessageWildMarshaller,	2765
activemq::wireformat::openwire::marshal::v5::MessagePurgeMarshaller,	2834
activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilteredMarshaller,	2890
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller,	3016
activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller,	3143
activemq::wireformat::openwire::marshal::v5::ProducerWildMarshaller,	3175
activemq::wireformat::openwire::marshal::v5::ProducerWildMarshaller,	3204
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller,	3306
activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller,	3336
activemq::wireformat::openwire::marshal::v5::ReplyCommandMarshaller,	3370
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller,	3400
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller,	3494
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller,	3514
activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller,	3590
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller,	3796
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller,	3965
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller,	4106
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller,	4170
activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller,	212
activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller,	253
activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller,	387
activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller,	415
activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller,	463
activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller,	509
activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller,	576
activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller,	634
activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller,	664
activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller,	690
activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller,	720
activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller,	896
activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller,	929
activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller,	1325
activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller,	1359
activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller,	1390
activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller,	1422
activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller,	1469

activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller, 3179  
 1499  
 activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller, 3213  
 1532  
 activemq::wireformat::openwire::marshal::v6::ControlConsumerMarshaller, 3293  
 1562  
 activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller, 3332  
 1597  
 activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller, 3365  
 1640  
 activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller, 3415  
 1803  
 activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller, 3490  
 1816  
 activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller, 3510  
 1899  
 activemq::wireformat::openwire::marshal::v6::FlushOwnerInfoMarshaller, 3577  
 2003  
 activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller, 3804  
 2164  
 activemq::wireformat::openwire::marshal::v6::JmsQueueInfoMarshaller, 3978  
 2229  
 activemq::wireformat::openwire::marshal::v6::JmsQueueTopicAckMarshaller, 4110  
 2270  
 activemq::wireformat::openwire::marshal::v6::JmsQueueTraceMarshaller, 4148  
 2293  
 activemq::wireformat::openwire::marshal::v6::JmsQueueTraceMarshaller, 4148  
 2312  
 activemq::wireformat::openwire::marshal::v6::JmsQueueTraceMarshaller, 4148  
 2340  
 activemq::wireformat::openwire::marshal::v6::JmsQueueTraceMarshaller, 4148  
 2375  
 activemq::wireformat::openwire::marshal::v6::JmsQueueTraceMarshaller, 4148  
 2426  
 activemq::wireformat::openwire::marshal::v6::JmsQueueTraceMarshaller, 4148  
 2650  
 activemq::wireformat::openwire::marshal::v6::JmsQueueTraceMarshaller, 4148  
 2713  
 activemq::wireformat::openwire::marshal::v6::JmsQueueTraceMarshaller, 4148  
 2722  
 activemq::wireformat::openwire::marshal::v6::JmsQueueTraceMarshaller, 4148  
 2773  
 activemq::wireformat::openwire::marshal::v6::JmsQueueTraceMarshaller, 4148  
 2851  
 activemq::wireformat::openwire::marshal::v6::JmsQueueTraceMarshaller, 4148  
 2886  
 activemq::wireformat::openwire::marshal::v6::JmsQueueTraceMarshaller, 4148  
 3007  
 activemq::wireformat::openwire::marshal::v6::JmsQueueTraceMarshaller, 4148  
 3147

- getDefaultDestinationName
  - activemq::cmsutil::CmsTemplate, 1207
- getDefaultSSLParameters
  - decaf::net::ssl::SSLContext, 3654
- getDelay
  - decaf::util::concurrent::Delayed, 1775
- getDeliveryMode
  - activemq::cmsutil::CachedProducer, 1103
  - activemq::cmsutil::CmsTemplate, 1207
  - activemq::core::ActiveMQProducer, 468
  - cms::MessageProducer, 2811
- getDeliverySequenceId
  - activemq::commands::MessageDispatchNotification, 2718
- getDestination
  - activemq::cmsutil::CmsTemplate::ProducerInfo, 3156
  - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3266
  - activemq::cmsutil::CmsTemplate::ResolveProducer, 3373
  - activemq::cmsutil::CmsTemplate::ResolveReceiver, 3374
  - activemq::commands::ConsumerControl, 1444, 1445
  - activemq::commands::ConsumerInfo, 1505, 1506
  - activemq::commands::DestinationInfo, 1782, 1783
  - activemq::commands::JournalQueueAck, 2226, 2227
  - activemq::commands::JournalTopicAck, 2254, 2255
  - activemq::commands::Message, 2604, 2605
  - activemq::commands::MessageAck, 2645, 2646
  - activemq::commands::MessageDispatch, 2681
  - activemq::commands::MessageDispatchNotification, 2719
  - activemq::commands::MessagePull, 2826, 2827
  - activemq::commands::ProducerInfo, 3188
  - activemq::commands::SubscriptionInfo, 3784, 3785
- getDestinationResolver
  - activemq::cmsutil::CmsDestinationAccessor, 1189
- getDestinationType
  - activemq::commands::ActiveMQDestination, 318
  - activemq::commands::ActiveMQQueue, 482
  - activemq::commands::ActiveMQTempQueue, 610
  - activemq::commands::ActiveMQTempTopic, 640
  - activemq::commands::ActiveMQTopic, 700
  - cms::Destination, 1778
- getDisableMessageID
  - activemq::cmsutil::CachedProducer, 1103
  - activemq::core::ActiveMQProducer, 469
  - cms::MessageProducer, 2811
- getDisableMessageTimeStamp
  - activemq::cmsutil::CachedProducer, 1104
  - activemq::core::ActiveMQProducer, 469
  - cms::MessageProducer, 2812
- getDoubleArray
  - activemq::commands::ActiveMQMapMessage, 356
- getDoubleAt
  - activemq::util::PrimitiveList, 3072
  - activemq::util::PrimitiveMap, 3084
  - activemq::util::PrimitiveValueNode, 3108
- getDoubleCapacity
  - decaf::internal::nio::ByteBuffer, 1026
  - decaf::internal::util::ByteArrayAdapter, 989
  - decaf::nio::ByteBuffer, 1063, 1064
- getDoubleProperty
  - decaf::internal::util::ByteArrayAdapter, 990
- getDoubleProperty
  - activemq::commands::ActiveMQMessageTemplate, 427
- getDurableTopicPrefetch
  - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2820
  - cms::Message, 2628
- getPrefetchPolicy
  - activemq::core::policies::DefaultPrefetchPolicy, 1728
  - activemq::core::PrefetchPolicy, 3065

- getEnabledCipherSuites
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2928
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2933
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2947
  - decaf::net::ssl::SSLServerSocket, 3665
  - decaf::net::ssl::SSLSocket, 3674
- getEnabledProtocols
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2929
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2933
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2948
  - decaf::net::ssl::SSLServerSocket, 3665
  - decaf::net::ssl::SSLSocket, 3674
- getEncoded
  - decaf::security::auth::x500::X500Principal, 4140
  - decaf::security::cert::Certificate, 1114
  - decaf::security::Key, 2365
- getEnumeration
  - activemq::core::ActiveMQQueueBrowser, 485
  - cms::QueueBrowser, 3244
- getenv
  - decaf::lang::System, 3843
- getErrorCode
  - decaf::net::SocketError, 3627
- getErrorMessage
  - decaf::util::logging::Handler, 2043
- getErrorMessage
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2958
  - decaf::net::SocketError, 3627
- getException
  - activemq::commands::ConnectionError, 1334, 1335
  - activemq::commands::ExceptionResponse, 1897
- getExceptionClass
  - activemq::commands::BrokerError, 871
- getExceptionListener
  - activemq::core::ActiveMQConnection, 270
  - activemq::core::ActiveMQConnectionFactory, 287
  - activemq::core::ActiveMQSession, 526
- cms::Connection, 1299
- SSLParameters, 2928
- activemq::commands::Message, 2605
- SSLServerSocket, 2933
- decaf::net::SocketImpl, 3640
- SSLSocket, 2947
- decaf::util::logging::Handler, 2043
- decaf::util::logging::Logger, 2468
- getFirstMessageId
  - activemq::commands::MessageAck, 2646
- SSLParameters, 2929
- activemq::commands::ReplayCommand, 3665
- getFloat
  - activemq::commands::ActiveMQMapMessage, 357
- activemq::util::PrimitiveList, 3072
- activemq::util::PrimitiveMap, 3085
- activemq::util::PrimitiveValueNode, 3109
- cms::MapMessage, 2556
- decaf::internal::nio::ByteBuffer, 1027
- decaf::internal::util::ByteArrayAdapter, 991
- decaf::nio::ByteBuffer, 1064, 1065
- getFloatArray
  - decaf::internal::util::ByteArrayAdapter, 991
- getFloatAt
  - decaf::internal::util::ByteArrayAdapter, 991
- getFloatCapacity
  - decaf::internal::util::ByteArrayAdapter, 992
- SSLParameters, 2958
- activemq::commands::ActiveMQMessageTemplate, 428
- activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2820
- cms::Message, 2628
- getFormat
  - decaf::security::Key, 2365
- getFormatId
  - activemq::commands::XATransactionId, 4146
- getFormatter
  - decaf::util::logging::Handler, 2044
- getFragment
  - activemq::util::CompositeData, 1254
  - decaf::internal::net::URIType, 4066

- decaf::net::URI, 4037
- getFreeThreadCount
  - decaf::util::concurrent::ThreadPool, 3892
- getGlobalPool
  - decaf::internal::AprPool, 736
  - decaf::internal::DecafRuntime, 1724
- getGlobalTransactionId
  - activemq::commands::XATransactionId, 4146
- getGroupID
  - activemq::commands::Message, 2605
- getGroupSequence
  - activemq::commands::Message, 2605
- getHandlers
  - decaf::util::logging::Logger, 2468
- getHead
  - decaf::util::logging::Formatter, 2028
  - decaf::util::logging::XMLFormatter, 4173
- getHoldCount
  - decaf::util::concurrent::locks::ReentrantLock, 3275
- getHost
  - activemq::util::CompositeData, 1254
  - decaf::internal::net::URIType, 4066
  - decaf::net::URI, 4037
- getHostAddress
  - decaf::net::InetAddress, 2081
- getHostName
  - decaf::net::InetAddress, 2081
- getHostname
  - activemq::util::IdGenerator, 2053
- getId
  - activemq::state::TransactionState, 3991
  - decaf::lang::Thread, 3881
- getIndex
  - decaf::net::URISyntaxException, 4063
- getInetAddress
  - decaf::net::Socket, 3616
  - decaf::net::SocketImpl, 3640
- getInfo
  - activemq::state::ConnectionState, 1431
  - activemq::state::ConsumerState, 1536
  - activemq::state::ProducerState, 3216
  - activemq::state::SessionState, 3537
- getInitialDelayTime
  - activemq::transport::inactivity::InactivityMonitor, 2067
- getInitialReconnectDelay
  - activemq::transport::failover::FailoverTransport, 1934
- getInitialRedeliveryDelay
  - activemq::core::policies::DefaultRedeliveryPolicy, 1732
  - activemq::core::RedeliveryPolicy, 3270
- getInput
  - decaf::net::URISyntaxException, 4063
- getInputStream
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2948
  - decaf::internal::net::tcp::TcpSocket, 3856
  - decaf::net::Socket, 3616
  - decaf::net::SocketImpl, 3640
- getInstance
  - activemq::transport::mock::MockTransport, 2857
  - activemq::transport::TransportRegistry, 4017
  - activemq::wireformat::WireFormatRegistry, 4131
  - decaf::util::concurrent::ThreadPool, 3892
  - decaf::util::logging::LogWriter, 2494
- getInt
  - activemq::commands::ActiveMQMapMessage, 357
  - activemq::util::PrimitiveList, 3073
  - activemq::util::PrimitiveMap, 3085
  - activemq::util::PrimitiveValueNode, 3109
  - cms::MapMessage, 2556
  - decaf::internal::nio::ByteBuffer, 1027, 1028
  - decaf::internal::util::ByteArrayAdapter, 992
  - decaf::nio::ByteBuffer, 1065
- getIntArray
  - decaf::internal::util::ByteArrayAdapter, 992
- getIntAt
  - decaf::internal::util::ByteArrayAdapter, 993
- getIntCapacity
  - decaf::internal::util::ByteArrayAdapter, 993
- getIntProperty
  - activemq::commands::ActiveMQMessageTemplate, 428
  - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2821
- getMessage
  - activemq::commands::Message, 2629
- getIssuerUniqueID

- decaf::security::cert::X509Certificate, 4142
- getIssuerX500Principal
  - decaf::security::cert::X509Certificate, 4142
- getKeepAlive
  - decaf::net::Socket, 3616
- getKey
  - decaf::util::Map::Entry, 1881
- getKeyUsage
  - decaf::security::cert::X509Certificate, 4142
- getLastDeliveredSequenceId
  - activemq::commands::RemoveInfo, 3286
  - activemq::core::ActiveMQConsumer, 306
  - activemq::core::ActiveMQSession, 526
- getLastMessageId
  - activemq::commands::MessageAck, 2646
- getLastNakNumber
  - activemq::commands::ReplayCommand, 3346
- getLastSequenceId
  - activemq::util::LongSequenceGenerator, 2535
- getLeastSignificantBits
  - decaf::util::UUID, 4084
- getLevel
  - decaf::util::logging::Handler, 2044
  - decaf::util::logging::Logger, 2468
  - decaf::util::logging::LogRecord, 2489
- getLimit
  - activemq::util::MemoryUsage, 2594
- getList
  - activemq::util::PrimitiveValueNode, 3109
- getLocalAddress
  - decaf::internal::net::tcp::TcpSocket, 3856
  - decaf::net::Socket, 3617
  - decaf::net::SocketImpl, 3640
- getLocalHost
  - decaf::net::InetAddress, 2082
- getLocalPort
  - decaf::net::ServerSocket, 3453
  - decaf::net::Socket, 3617
  - decaf::net::SocketImpl, 3641
- getLogger
  - decaf::util::logging::Logger, 2469
  - decaf::util::logging::LogManager, 2484
- getLoggerName
  - decaf::util::logging::LogRecord, 2489
- getLoggerNames
  - decaf::util::logging::LogManager, 2484
- getLogManager
  - decaf::util::logging::LogManager, 2485
- getLong
  - activemq::commands::ActiveMQMapMessage, 358
  - activemq::util::PrimitiveList, 3073
  - activemq::util::PrimitiveMap, 3085
  - activemq::util::PrimitiveValueNode, 3109
  - cms::MapMessage, 2557
  - decaf::internal::nio::ByteBuffer, 1028
  - decaf::internal::util::ByteArrayAdapter, 993
  - decaf::nio::ByteBuffer, 1066
- getLongArray
  - decaf::internal::util::ByteArrayAdapter, 994
- getLongAt
  - decaf::internal::util::ByteArrayAdapter, 994
- getLongCapacity
  - decaf::internal::util::ByteArrayAdapter, 994
- getLongProperty
  - activemq::commands::ActiveMQMessageTemplate, 429
  - activemq::wireformat::openwire::utils::MessagePropertyInter, 2821
  - cms::Message, 2629
- getLoopbackAddress
  - decaf::net::InetAddress, 2082
- getMagic
  - activemq::commands::WireFormatInfo, 4098
- getManaged
  - decaf::internal::util::GenericResource, 2038
- getMap
  - activemq::commands::ActiveMQMapMessage, 358
  - activemq::util::PrimitiveValueNode, 3110
- getMapNames
  - activemq::commands::ActiveMQMapMessage, 358
  - cms::MapMessage, 2557
- getMarshaledForm
  - activemq::commands::BaseDataStructure, 840



- activemq::wireformat::MarshalAware, 2566
- getMarshaledProperties
  - activemq::commands::Message, 2605
  - activemq::commands::WireFormatInfo, 4098
- getMaxCacheSize
  - activemq::state::ConnectionStateTracker, 1435
  - activemq::transport::failover::FailoverTransport, 1934
- getMaximumPendingMessageLimit
  - activemq::commands::ConsumerInfo, 1506
- getMaximumRedeliveries
  - activemq::core::policies::DefaultRedeliveryPolicy, 1732
  - activemq::core::RedeliveryPolicy, 3270
- getMaxInactivityDuration
  - activemq::commands::WireFormatInfo, 4098
  - activemq::wireformat::openwire::OpenWireFormatInfo, 2976
- getMaxInactivityDurationInitialDelay
  - activemq::commands::WireFormatInfo, 4098
  - activemq::wireformat::openwire::OpenWireFormatInfo, 2976
- getMaxPrefetchLimit
  - activemq::core::policies::DefaultPrefetchPolicy, 1728
  - activemq::core::PrefetchPolicy, 3065
- getMaxReconnectAttempts
  - activemq::transport::failover::FailoverTransport, 1934
- getMaxReconnectDelay
  - activemq::transport::failover::FailoverTransport, 1934
- getMaxThreads
  - decaf::util::concurrent::ThreadPool, 3892
- getMessage
  - activemq::cmsutil::CmsTemplate::ReceiveFile, 3267
  - activemq::commands::BrokerError, 871
  - activemq::commands::JournalTrace, 2283
  - activemq::commands::MessageDispatch, 2681
  - activemq::core::DispatchData, 1840
  - cms::CMSException, 1192
  - decaf::lang::Exception, 1890
  - decaf::lang::Throwable, 3897
  - decaf::util::logging::LogRecord, 2489
  - getMessageAck
    - activemq::commands::JournalQueueAck, 2227
  - getMessageAvailableCount
    - activemq::core::ActiveMQConsumer, 306
  - getMessageCount
    - activemq::commands::MessageAck, 2646
  - getMessageId
    - activemq::commands::JournalTopicAck, 2255
    - activemq::commands::Message, 2605
    - activemq::commands::MessageDispatchNotification, 2719
    - activemq::commands::MessagePull, 2827
  - getMessageListener
    - activemq::cmsutil::CachedConsumer, 1099
    - activemq::core::ActiveMQConsumer, 306
  - getMessageProperties
    - activemq::commands::Message, 2605
  - getMessageSelector
    - activemq::cmsutil::CachedConsumer, 1099
    - activemq::core::ActiveMQConsumer, 307
  - getMessageSequenceId
    - activemq::core::ActiveMQQueueBrowser, 485
    - cms::MessageConsumer, 2675
    - cms::QueueBrowser, 3244
  - activemq::commands::JournalTopicAck, 2255
  - getMetaData
    - activemq::core::ActiveMQConnection, 271
  - activemq::core::ActiveMQConnection, 1299
  - activemq::commands::ActiveMQBlobMessage, 187
  - getMostSignificantBits
    - decaf::util::UUID, 4084
  - getName
    - activemq::commands::ActiveMQBlobMessage, 187

decaf::lang::Thread, 3882  
 decaf::security::auth::x500::X500Principal, 4140  
 decaf::security::Principal, 3115  
 decaf::util::logging::Level, 2406  
 decaf::util::logging::Logger, 2469  
 getNeedClientAuth  
 decaf::internal::net::ssl::openssl::OpenSSLParameters, 2858  
 decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2858  
 decaf::internal::net::ssl::openssl::OpenSSLSocket, 2858  
 decaf::internal::net::ssl::openssl::OpenSSLSocket, 2858  
 decaf::net::ssl::SSLParameters, 3660  
 decaf::net::ssl::SSLServerSocket, 3665  
 decaf::net::ssl::SSLSocket, 3675  
 getNetworkBrokerId  
 activemq::commands::NetworkBridgeFilter, 2879, 2880  
 getNetworkConsumerPath  
 activemq::commands::ConsumerInfo, 1506  
 getNetworkProperties  
 activemq::commands::BrokerInfo, 906, 907  
 getNetworkRuntime  
 decaf::internal::net::Network, 2876  
 getNetworkTTL  
 activemq::commands::NetworkBridgeFilter, 2880  
 getNextConsumerId  
 activemq::core::ActiveMQSession, 526  
 getNextLocalTransactionId  
 activemq::core::ActiveMQConnection, 271  
 getNextProducerId  
 activemq::core::ActiveMQSession, 526  
 getNextSequenceId  
 activemq::util::LongSequenceGenerator, 2535  
 getNextSessionId  
 activemq::core::ActiveMQConnection, 271  
 getNextTempDestinationId  
 activemq::core::ActiveMQConnection, 271  
 getNotAfter  
 decaf::security::cert::X509Certificate, 4142  
 getNotBefore  
 decaf::security::cert::X509Certificate, 4142  
 getNumReceivedMessageBeforeFail  
 activemq::transport::mock::MockTransport, 2858  
 getNumReceivedMessages  
 activemq::transport::mock::MockTransport, 2858  
 getNumSentKeepAlives  
 activemq::transport::mock::MockTransport, 2858  
 getNumSentKeepAlivesBeforeFail  
 activemq::transport::mock::MockTransport, 2858  
 getNumSentMessageBeforeFail  
 activemq::transport::mock::MockTransport, 2858  
 getNumSentMessages  
 activemq::transport::mock::MockTransport, 2858  
 getObjectId  
 activemq::commands::RemoveInfo, 3286  
 getOOBInline  
 decaf::net::Socket, 3617  
 getOperationType  
 activemq::commands::DestinationInfo, 1783  
 getOption  
 decaf::internal::net::tcp::TcpSocket, 3856  
 decaf::net::SocketImpl, 3641  
 getOptions  
 activemq::commands::ActiveMQDestination, 318  
 getOrderedTarget  
 activemq::commands::ActiveMQDestination, 318  
 getOriginalDestination  
 activemq::commands::Message, 2606  
 getOriginalTransactionId  
 activemq::commands::Message, 2606  
 getOutputStream  
 decaf::internal::net::ssl::openssl::OpenSSLSocket, 2949  
 decaf::internal::net::tcp::TcpSocket, 3857  
 decaf::net::Socket, 3617  
 decaf::net::SocketImpl, 3641  
 getParameters  
 activemq::util::CompositeData, 1254  
 getParent  
 decaf::util::logging::Logger, 2469

- getParentId
  - activemq::commands::ConsumerId, 1475
  - activemq::commands::ProducerId, 3160
  - activemq::commands::SessionId, 3479
- getPassword
  - activemq::commands::ConnectionInfo, 1396, 1397
  - activemq::core::ActiveMQConnection, 271
  - activemq::core::ActiveMQConnectionFactory, 287
- getPath
  - activemq::util::CompositeData, 1254
  - decaf::internal::net::URIType, 4066
  - decaf::net::URI, 4037
- getPeerBrokerInfos
  - activemq::commands::BrokerInfo, 907
- getPhysicalName
  - activemq::commands::ActiveMQDestination, 318
- getPooledThreadListener
  - decaf::util::concurrent::PooledThread, 3057
- getPoolSize
  - decaf::util::concurrent::ThreadPool, 3892
- getPort
  - decaf::internal::net::URIType, 4066
  - decaf::net::Socket, 3618
  - decaf::net::SocketImpl, 3642
  - decaf::net::URI, 4037
- getPreferredWireFormatInfo
  - activemq::wireformat::openwire::OpenWireFormat, 2976
- getPrefetch
  - activemq::commands::ConsumerControl, 1445
- getPrefetchPolicy
  - activemq::core::ActiveMQConnection, 272
  - activemq::core::ActiveMQConnectionFactory, 287
- getPrefetchSize
  - activemq::commands::ConsumerInfo, 1506
- getPreparedResult
  - activemq::state::TransactionState, 3991
- getPriority
  - activemq::cmsutil::CachedProducer, 1194
  - activemq::cmsutil::CmsTemplate, 1207
- activemq::commands::ConsumerInfo, 1506
- activemq::commands::Message, 2606
- activemq::core::ActiveMQProducer, 469
- cms::MessageProducer, 2812
- decaf::lang::Thread, 3882
- getProducerId
  - activemq::commands::Message, 2606
  - activemq::commands::MessageId, 2753, 2754
  - activemq::commands::ProducerAck, 3127
  - activemq::commands::ProducerInfo, 3188
  - activemq::core::ActiveMQProducer, 469
- getProducerInfo
  - activemq::core::ActiveMQProducer, 470
- getProducerSequenceId
  - activemq::commands::MessageId, 2754
- getProducerState
  - activemq::state::SessionState, 3537
- getProducerStates
  - activemq::state::SessionState, 3537
  - activemq::state::TransactionState, 3991
- getProducerWindowSize
  - activemq::core::ActiveMQConnection, 272
  - activemq::core::ActiveMQConnectionFactory, 288
- getProperties
  - activemq::commands::WireFormatInfo, 4099
  - activemq::util::ActiveMQProperties, 476, 477
  - activemq::wireformat::stomp::StompFrame, 3744
  - decaf::lang::System, 3843
  - decaf::util::logging::LogManager, 2485
- getProperty
  - activemq::util::ActiveMQProperties, 477
  - activemq::wireformat::stomp::StompFrame, 3744
  - cms::CMSProperties, 1197
  - decaf::lang::System, 3844
  - decaf::util::logging::LogManager, 2485
  - decaf::util::Properties, 3220
- getPropertyNames
  - activemq::commands::ActiveMQMessageTemplate, 429
  - cms::Message, 2630
- getProtocols
  - decaf::net::ssl::SSLParameters, 3660

- getProviderMajorVersion
  - activemq::core::ActiveMQConnectionMetaData, 2067
  - 296
  - cms::ConnectionMetaData, 1428
- getProviderMinorVersion
  - activemq::core::ActiveMQConnectionMetaData, 2067
  - 296
  - cms::ConnectionMetaData, 1428
- getProviderVersion
  - activemq::core::ActiveMQConnectionMetaData, 2067
  - 296
  - cms::ConnectionMetaData, 1429
- getPublicKey
  - decaf::security::cert::Certificate, 1114
- getQuery
  - decaf::internal::net::URIType, 4067
  - decaf::net::URI, 4037
- getQueue
  - activemq::core::ActiveMQQueueBrowser, 485
  - cms::QueueBrowser, 3245
- getQueueBrowserPrefetch
  - activemq::core::policies::DefaultPrefetchPolicy, 1728
  - activemq::core::PrefetchPolicy, 3065
- getQueueName
  - activemq::commands::ActiveMQQueue, 483
  - activemq::commands::ActiveMQTempQueue, 610
  - cms::Queue, 3239
  - cms::TemporaryQueue, 3872
- getQueuePrefetch
  - activemq::core::policies::DefaultPrefetchPolicy, 1728
  - activemq::core::PrefetchPolicy, 3065
- getRawAuthority
  - decaf::net::URI, 4037
- getRawFragment
  - decaf::net::URI, 4038
- getRawPath
  - decaf::net::URI, 4038
- getRawQuery
  - decaf::net::URI, 4038
- getRawSchemeSpecificPart
  - decaf::net::URI, 4038
- getRawUserInfo
  - decaf::net::URI, 4039
- getReadCheckTime
  - activemq::transport::inactivity::InactivityMonitor, 2067
  - getReason
    - decaf::net::URISyntaxException, 4063
  - getReceiveBufferSize
    - decaf::net::ServerSocket, 3453
    - decaf::net::Socket, 3618
  - getReceiveTimeout
    - activemq::cmsutil::CmsTemplate, 1207
  - getReconnectDelay
    - activemq::transport::failover::FailoverTransport, 1934
  - getReconnectTo
    - activemq::commands::ConnectionControl, 1304, 1305
  - getRecoveringPullConsumers
    - activemq::state::ConnectionState, 1431
  - getRedeliveryCounter
    - activemq::commands::Message, 2606
    - activemq::commands::MessageDispatch, 2681
  - getRedeliveryDelay
    - activemq::core::policies::DefaultRedeliveryPolicy, 1732
    - activemq::core::RedeliveryPolicy, 3270
  - getRedeliveryPolicy
    - activemq::core::ActiveMQConnection, 272
    - activemq::core::ActiveMQConnectionFactory, 288
    - activemq::core::ActiveMQConsumer, 307
  - getRemaining
    - decaf::util::zip::Inflater, 2091
  - getRemoteAddress
    - activemq::transport::failover::FailoverTransport, 1934
    - activemq::transport::IOTransport, 2216
    - activemq::transport::mock::MockTransport, 2858
    - activemq::transport::Transport, 3998
    - activemq::transport::TransportFilter, 4008
  - getRemoteBlobUrl
    - activemq::commands::ActiveMQBlobMessage, 187
  - getReplyTo
    - activemq::commands::Message, 2606
  - getResourceLifecycleManager
    - activemq::cmsutil::CmsAccessor, 1186
    - activemq::cmsutil::SessionPool, 3535

- getResponse
  - activemq::transport::correlator::FutureResponse, 2033, 2034
- getResult
  - activemq::commands::IntegerResponse, 2162
- getReuseAddress
  - decaf::net::ServerSocket, 3454
  - decaf::net::Socket, 3618
- getRuntime
  - decaf::lang::Runtime, 3420
- getRuntimeLock
  - decaf::internal::net::Network, 2876
- getSafeValue
  - decaf::util::StlQueue, 3726
- getScheme
  - activemq::util::CompositeData, 1254
  - decaf::internal::net::URIType, 4067
  - decaf::net::URI, 4039
- getSchemeSpecificPart
  - decaf::internal::net::URIType, 4067
  - decaf::net::URI, 4039
- getSeedFromId
  - activemq::util::IdGenerator, 2053
- getSelector
  - activemq::commands::ConsumerInfo, 1506
  - activemq::commands::SubscriptionInfo, 3785
- getSendBufferSize
  - decaf::net::Socket, 3618
- getSendTimeout
  - activemq::core::ActiveMQConnection, 272
  - activemq::core::ActiveMQConnectionFactory, 288
  - activemq::core::ActiveMQProducer, 470
- getSequenceFromId
  - activemq::util::IdGenerator, 2053
- getServerSocketFactory
  - decaf::net::ssl::SSLContext, 3654
- getServiceName
  - activemq::commands::DiscoveryEvent, 1814
- getSession
  - activemq::cmsutil::PooledSession, 3054
- getSessionAcknowledgeMode
  - activemq::cmsutil::CmsAccessor, 1186
- getSessionId
  - activemq::commands::ConsumerId, 1476
- activemq::commands::ProducerId, 3160
- activemq::commands::SessionInfo, 3507
- activemq::core::ActiveMQSession, 526
- getSessionInfo
  - activemq::core::ActiveMQSession, 527
- getSessionState
  - activemq::state::ConnectionState, 1431
- getSessionStates
  - activemq::state::ConnectionState, 1431
- getShort
  - activemq::commands::ActiveMQMapMessage, 359
  - activemq::util::PrimitiveList, 3074
  - activemq::util::PrimitiveMap, 3086
  - activemq::util::PrimitiveValueNode, 3110
  - cms::MapMessage, 2557
  - decaf::internal::nio::ByteBuffer, 1029
  - decaf::internal::util::ByteArrayAdapter, 994
  - decaf::nio::ByteBuffer, 1067
- getShortArray
  - decaf::internal::util::ByteArrayAdapter, 995
- getShortAt
  - decaf::internal::util::ByteArrayAdapter, 995
- getShortCapacity
  - decaf::internal::util::ByteArrayAdapter, 996
- getShortProperty
  - activemq::commands::ActiveMQMessageTemplate, 430
  - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2821
  - cms::Message, 2630
- getSigAlgName
  - decaf::security::cert::X509Certificate, 4142
- getSigAlgOID
  - decaf::security::cert::X509Certificate, 4142
- getSigAlgParams
  - decaf::security::cert::X509Certificate, 4142
- getSignature
  - decaf::security::cert::X509Certificate, 4142
- getSize
  - activemq::commands::ConsumerId, 1476

activemq::commands::ActiveMQTextMessage, 2631  
 670  
 activemq::commands::Message, 2606  
 activemq::commands::ProducerAck, 3127  
 getSocketFactory  
 decaf::net::ssl::SSLContext, 3654  
 getSocketHandle  
 decaf::internal::net::tcp::TcpSocket, 3857  
 getSoLinger  
 decaf::net::Socket, 3619  
 getSoTimeout  
 decaf::net::ServerSocket, 3454  
 decaf::net::Socket, 3619  
 getSource  
 decaf::internal::net::URIType, 4067  
 getSourceFile  
 decaf::util::logging::LogRecord, 2489  
 getSourceFunction  
 decaf::util::logging::LogRecord, 2490  
 getSourceLine  
 decaf::util::logging::LogRecord, 2490  
 getSSLParameters  
 decaf::net::ssl::SSLSocket, 3675  
 getStackTrace  
 cms::CMSEException, 1193  
 decaf::lang::Exception, 1891  
 decaf::lang::Throwable, 3897  
 getStackTraceElements  
 activemq::commands::BrokerError, 871  
 getStackTraceString  
 cms::CMSEException, 1193  
 decaf::lang::Exception, 1891  
 decaf::lang::Throwable, 3898  
 getStartupMaxReconnectAttempts  
 activemq::transport::failover::FailoverTransport, 1934  
 getState  
 decaf::lang::Thread, 3882  
 getString  
 activemq::commands::ActiveMQMapMessage, 359  
 activemq::util::PrimitiveList, 3074  
 activemq::util::PrimitiveMap, 3086  
 activemq::util::PrimitiveValueNode, 3110  
 cms::MapMessage, 2558  
 getStringProperty  
 activemq::commands::ActiveMQMessageText, 430  
 activemq::wireformat::openwire::utils::MessageProperty, 2821  
 cms::Message, 2631  
 getSubscriptionName  
 activemq::commands::RemoveSubscriptionInfo, 3316  
 activemq::commands::SubscriptionInfo, 3785  
 getSubjectUniqueID  
 decaf::security::cert::X509Certificate, 4142  
 getSubjectX500Principal  
 decaf::security::cert::X509Certificate, 4142  
 getSubscribedDestination  
 activemq::commands::SubscriptionInfo, 3785  
 getSubscriptionName  
 activemq::commands::ConsumerInfo, 1506  
 getSubscriptionName  
 activemq::commands::JournalTopicAck, 2255  
 getSupportedCipherSuites  
 decaf::internal::net::ssl::DefaultSSLServerSocketFactory, 1749  
 decaf::internal::net::ssl::DefaultSSLSocketFactory, 1756  
 decaf::internal::net::ssl::openssl::OpenSSLParameters, 2929  
 decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2933  
 decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2940  
 decaf::internal::net::ssl::openssl::OpenSSLSocket, 2949  
 decaf::internal::net::ssl::openssl::OpenSSLSocketFactory, 2965  
 decaf::net::ssl::SSLServerSocket, 3665  
 decaf::net::ssl::SSLServerSocketFactory, 3669  
 decaf::net::ssl::SSLSocket, 3675  
 decaf::net::ssl::SSLSocketFactory, 3681  
 getSupportedProtocols  
 decaf::internal::net::ssl::openssl::OpenSSLParameters, 2929  
 decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2933  
 decaf::internal::net::ssl::openssl::OpenSSLSocket, 2949  
 decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 3666  
 decaf::net::ssl::SSLSocket, 3675

- getSupportedSSLParameters
  - decaf::net::ssl::SSLContext, 3655
- getTail
  - decaf::util::logging::Formatter, 2029
  - decaf::util::logging::XMLFormatter, 4174
- getTargetConsumerId
  - activemq::commands::Message, 2606, 2607
- getTBCertificate
  - decaf::security::cert::X509Certificate, 4142
- getTcpNoDelay
  - decaf::net::Socket, 3619
- getTempDestinations
  - activemq::state::ConnectionState, 1431
- getText
  - activemq::commands::ActiveMQTextMessage, 670
  - cms::TextMessage, 3875
- getThrown
  - decaf::util::logging::LogRecord, 2490
- getTime
  - decaf::util::Date, 1721
- getTimeout
  - activemq::commands::DestinationInfo, 1783
  - activemq::commands::MessagePull, 2827
  - activemq::transport::failover::FailoverTransport, 1935
- getTimestamp
  - activemq::commands::Message, 2607
  - decaf::util::logging::LogRecord, 2490
- getTimeToLive
  - activemq::cmsutil::CachedProducer, 1104
  - activemq::cmsutil::CmsTemplate, 1207
  - activemq::core::ActiveMQProducer, 470
  - cms::MessageProducer, 2813
- getTopicName
  - activemq::commands::ActiveMQTempTopic, 640
  - activemq::commands::ActiveMQTopic, 700
  - cms::TemporaryTopic, 3874
  - cms::Topic, 3931
- getTopicPrefetch
  - activemq::core::policies::DefaultPrefetchPolicy, 1728
  - activemq::core::PrefetchPolicy, 3065
- getTrafficClass
  - decaf::net::Socket, 3620
- getTransactionContext
  - activemq::core::ActiveMQSession, 527
- getTransactionId
  - activemq::commands::JournalTopicAck, 2255
  - activemq::commands::JournalTransaction, 2310
  - activemq::commands::Message, 2607
  - activemq::commands::MessageAck, 2646
  - activemq::commands::TransactionInfo, 3962
  - activemq::core::ActiveMQTransactionContext, 729
- getTransactionState
  - activemq::state::ConnectionState, 1431
  - activemq::state::ProducerState, 3216
- getTransactionStates
  - activemq::state::ConnectionState, 1431
- getTransport
  - activemq::core::ActiveMQConnection, 272
  - activemq::transport::failover::BackupTransport, 760
- getTransportListener
  - activemq::transport::failover::FailoverTransport, 1935
  - activemq::transport::IOTransport, 2216
  - activemq::transport::mock::MockTransport, 2858
  - activemq::transport::Transport, 3998
  - activemq::transport::TransportFilter, 4008
- getTransportNames
  - activemq::transport::TransportRegistry, 4017
- getTreadId
  - decaf::util::logging::LogRecord, 2490
- getType
  - activemq::commands::JournalTransaction, 2310
  - activemq::commands::Message, 2607
  - activemq::commands::TransactionInfo, 3962
  - activemq::util::PrimitiveValueNode, 3111
  - decaf::security::cert::Certificate, 1114
- getUncaughtExceptionHandler
  - decaf::lang::Thread, 3882
- getUnconsumedMessages
  - activemq::core::ActiveMQSessionExecutor, 534
- getURI

activemq::transport::failover::URIPool, 4056  
 getUri  
 activemq::transport::failover::BackupTransport, 760  
 getUsage  
 activemq::util::MemoryUsage, 2594  
 getUseClientMode  
 decaf::internal::net::ssl::openssl::OpenSSLParameters, 2929  
 decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2934  
 decaf::internal::net::ssl::openssl::OpenSSLSocket, 2950  
 decaf::net::ssl::SSLParameters, 3660  
 decaf::net::ssl::SSLServerSocket, 3666  
 decaf::net::ssl::SSLSocket, 3676  
 getUseParentHandlers  
 decaf::util::logging::Logger, 2469  
 getUserID  
 activemq::commands::Message, 2607  
 getUserInfo  
 decaf::internal::net::URIType, 4067  
 decaf::net::URI, 4039  
 getUserName  
 activemq::commands::ConnectionInfo, 1397  
 getUsername  
 activemq::core::ActiveMQConnection, 273  
 activemq::core::ActiveMQConnectionFactory, 288  
 getValue  
 activemq::commands::BrokerId, 877  
 activemq::commands::ConnectionId, 1367, 1368  
 activemq::commands::ConsumerId, 1476  
 activemq::commands::LocalTransactionId, 2423  
 activemq::commands::ProducerId, 3160  
 activemq::commands::SessionId, 3480  
 activemq::util::PrimitiveValueNode, 3114  
 decaf::internal::net::SocketFileDescriptor, 3635  
 decaf::util::Map::Entry, 1881  
 decaf::util::zip::Adler32, 731  
 decaf::util::zip::Checksum, 1175  
 decaf::util::zip::CRC32, 1569  
 getVersion  
 activemq::commands::WireFormatInfo, 4099  
 activemq::wireformat::openwire::OpenWireFormat, 2976  
 activemq::wireformat::stomp::StompWireFormat, 3752  
 activemq::wireformat::WireFormat, 4090  
 decaf::security::cert::X509Certificate, 4142  
 getWantClientAuth  
 decaf::internal::net::ssl::openssl::OpenSSLParameters, 2929  
 decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2934  
 decaf::internal::net::ssl::openssl::OpenSSLSocket, 2950  
 decaf::net::ssl::SSLParameters, 3660  
 decaf::net::ssl::SSLServerSocket, 3666  
 decaf::net::ssl::SSLSocket, 3676  
 getWasPrepared  
 activemq::commands::JournalTransaction, 2310  
 getWhen  
 decaf::util::TimerTask, 3915  
 getWindowSize  
 activemq::commands::ProducerInfo, 3188  
 getWireFormat  
 activemq::transport::mock::MockTransport, 2858  
 getWireFormatNames  
 activemq::wireformat::WireFormatRegistry, 4131  
 getWriteCheckTime  
 activemq::transport::inactivity::InactivityMonitor, 2067  
 globalTransactionId  
 activemq::commands::XATransactionId, 4147  
 good\_match  
 internal\_state, 2191  
 groupID  
 activemq::commands::Message, 2613  
 groupSequence  
 activemq::commands::Message, 2613  
 GT\_OFF  
 gzguts.h, 4625  
 GUNZIP  
 inflate.h, 4626  
 GZ\_APPEND  
 gzguts.h, 4625  
 gz\_header  
 zlib.h, 4636  
 WireFormat, 2038  
 comm\_max, 2039



- comment, 2039
- done, 2039
- extra, 2039
- extra\_len, 2039
- extra\_max, 2039
- hcrc, 2039
- name, 2039
- name\_max, 2039
- os, 2039
- text, 2039
- time, 2039
- xflags, 2039
- gz\_headerp
  - zlib.h, 4636
- GZ\_NONE
  - gzguts.h, 4625
- GZ\_READ
  - gzguts.h, 4625
- gz\_state, 2039
  - direct, 2041
  - eof, 2041
  - err, 2041
  - fd, 2041
  - have, 2041
  - how, 2041
  - in, 2041
  - level, 2041
  - mode, 2041
  - msg, 2041
  - next, 2041
  - out, 2041
  - path, 2041
  - pos, 2041
  - raw, 2041
  - seek, 2041
  - size, 2041
  - skip, 2041
  - start, 2041
  - strategy, 2041
  - strm, 2041
  - want, 2041
- gz\_statep
  - gzguts.h, 4625
- GZ\_WRITE
  - gzguts.h, 4625
- GZBUFSIZE
  - gzguts.h, 4625
- gzFile
  - zlib.h, 4636
- gzguts.h
- COPY, 4625
- GT\_OFF, 4625
- GZ\_APPEND, 4625
- GZ\_NONE, 4625
- GZ\_READ, 4625
- gz\_statep, 4625
- GZ\_WRITE, 4625
- GZBUFSIZE, 4625
- GZIP, 4625
- local, 4625
- LOOK, 4625
- OF, 4625
- ZLIB\_INTERNAL, 4625
- zstrerror, 4625
- gzhead
  - internal\_state, 2191
- gzindex
  - internal\_state, 2191
- GZIP
  - deflate.h, 4623
  - gzguts.h, 4625
- Handler
  - decaf::util::logging::Handler, 2043
- handleTransportFailure
  - activemq::transport::failover::FailoverTransport, 1935
- hasArray
  - decaf::internal::nio::ByteBuffer, 1030
  - decaf::internal::nio::CharArrayBuffer, 1144
  - decaf::internal::nio::DoubleArrayBuffer, 1862
  - decaf::internal::nio::FloatArrayBuffer, 1983
  - decaf::internal::nio::IntArrayBuffer, 2128
  - decaf::internal::nio::LongArrayBuffer, 2520
  - decaf::internal::nio::ShortArrayBuffer, 3557
  - decaf::nio::ByteBuffer, 1067
  - decaf::nio::CharBuffer, 1159
  - decaf::nio::DoubleBuffer, 1873
  - decaf::nio::FloatBuffer, 1993
  - decaf::nio::IntBuffer, 2138
  - decaf::nio::LongBuffer, 2530
  - decaf::nio::ShortBuffer, 3568
- hash\_bits
  - internal\_state, 2191

- hash\_mask
  - internal\_state, 2191
- hash\_shift
  - internal\_state, 2191
- hash\_size
  - internal\_state, 2191
- hashCode
  - decaf::security::auth::x500::X500Principal, inflate.h, 4626
  - 4140
- hasMoreMessages
  - activemq::core::ActiveMQQueueBrowser, inflate\_state, 2087
  - 486
- hasMoreTokens
  - decaf::util::StringTokenizer, 3780
- hasNegotiator
  - activemq::wireformat::openwire::OpenWireFormat, 2977
- activemq::wireformat::stomp::StompWireFormat, 3752
  - activemq::wireformat::WireFormat, 4090
- hasNext
  - decaf::util::Iterator, 2223
- hasPrevious
  - decaf::util::ListIterator, 2418
- hasProperty
  - activemq::util::ActiveMQProperties, 477
  - activemq::wireformat::stomp::StompFrame, activemq::wireformat::stomp::StompCommandConstants, 3744
  - cms::CMSProperties, 1198
  - decaf::util::Properties, 3220
- hasRemaining
  - decaf::nio::Buffer, 940
- hasUnconsumedMessages
  - activemq::core::ActiveMQSessionExecutor, 534
- have
  - gz\_state, 2041
  - inflate\_state, 2087
- HAVE\_PTHREAD\_H
  - activemq/util/Config.h, 4275
  - decaf/util/Config.h, 4275
- HAVE\_UUID\_T
  - activemq/util/Config.h, 4275
  - decaf/util/Config.h, 4275
- HAVE\_UUID\_UUID\_H
  - activemq/util/Config.h, 4275
  - decaf/util/Config.h, 4275
- havedict
  - inflate\_state, 2087
- HCRC
  - inflate.h, 4627
- hcrc
  - gz\_header\_s, 2039
- HCRC\_STATE
  - deflate.h, 4623
- HEAD
  - head
- HEADER\_ACK
  - activemq::wireformat::stomp::StompCommandConstants, 3740
- HEADER\_CLIENT\_ID
  - activemq::wireformat::stomp::StompCommandConstants, 3740
- HEADER\_CONSUMERPRIORITY
  - activemq::wireformat::stomp::StompCommandConstants, 3740
- HEADER\_CONTENTLENGTH
  - activemq::wireformat::stomp::StompCommandConstants, 3740
- HEADER\_CORRELATIONID
  - activemq::wireformat::stomp::StompCommandConstants, 3740
- HEADER\_DESTINATION
  - activemq::wireformat::stomp::StompCommandConstants, 3740
- HEADER\_DISPATCH\_ASYNC
  - activemq::wireformat::stomp::StompCommandConstants, 3740
- HEADER\_EXCLUSIVE
  - activemq::wireformat::stomp::StompCommandConstants, 3740
- HEADER\_EXPIRES
  - activemq::wireformat::stomp::StompCommandConstants, 3740
- HEADER\_ID
  - activemq::wireformat::stomp::StompCommandConstants, 3740
- HEADER\_JMSPRIORITY
  - activemq::wireformat::stomp::StompCommandConstants, 3740
- HEADER\_LOGIN
  - activemq::wireformat::stomp::StompCommandConstants, 3740
- HEADER\_MAXPENDINGMSGLIMIT
  - activemq::wireformat::stomp::StompCommandConstants, 3740

HEADER\_MESSAGE activemq::wireformat::stomp::StompCommandConstants,  
 activemq::wireformat::stomp::StompCommandConstants, 3740  
 3740 HEADER\_SUBSCRIPTION  
 HEADER\_MESSAGEID activemq::wireformat::stomp::StompCommandConstants,  
 activemq::wireformat::stomp::StompCommandConstants, 3740  
 3740 HEADER\_SUBSCRIPTIONNAME  
 HEADER\_NOLOCAL activemq::wireformat::stomp::StompCommandConstants,  
 activemq::wireformat::stomp::StompCommandConstants, 3740  
 3740 HEADER\_TIMESTAMP  
 HEADER\_OLDSUBSCRIPTIONNAME activemq::wireformat::stomp::StompCommandConstants,  
 activemq::wireformat::stomp::StompCommandConstants, 3740  
 3740 HEADER\_TRANSACTIONID  
 HEADER\_PASSWORD activemq::wireformat::stomp::StompCommandConstants,  
 activemq::wireformat::stomp::StompCommandConstants, 3740  
 3740 HEADER\_TRANSFORMATION  
 HEADER\_PERSISTENT activemq::wireformat::stomp::StompCommandConstants,  
 activemq::wireformat::stomp::StompCommandConstants, 3740  
 3740 HEADER\_TRANSFORMATION\_ERROR  
 HEADER\_PREFETCHSIZE activemq::wireformat::stomp::StompCommandConstants,  
 activemq::wireformat::stomp::StompCommandConstants, 3740  
 3740 HEADER\_TYPE  
 HEADER\_RECEIPT\_REQUIRED activemq::wireformat::stomp::StompCommandConstants,  
 activemq::wireformat::stomp::StompCommandConstants, 3740  
 3740 heap  
 HEADER\_RECEIPTID internal\_state, 2191  
 activemq::wireformat::stomp::StompCommandConstants, heap\_len  
 3740 internal\_state, 2191  
 heap\_max  
 HEADER\_REDELIVERED internal\_state, 2191  
 activemq::wireformat::stomp::StompCommandConstants, 3740  
 3740 HEAP\_SIZE  
 HEADER\_REDELIVERYCOUNT deflate.h, 4623  
 activemq::wireformat::stomp::StompCommandConstants, HexStringParser  
 3740 decaf::internal::util::HexStringParser,  
 2047  
 HEADER\_REPLYTO HexTable  
 activemq::wireformat::stomp::StompCommandConstants, activemq::wireformat::openwire::utils::HexTable,  
 3740 2048  
 HEADER\_REQUESTID high\_water  
 activemq::wireformat::stomp::StompCommandConstants, internal\_state, 2191  
 3740 highestOneBit  
 HEADER\_RESPONSEID decaf::lang::Integer, 2149  
 activemq::wireformat::stomp::StompCommandConstants, decaf::lang::Long, 2501  
 3740 hold  
 HEADER\_RETROACTIVE inflate\_state, 2087  
 activemq::wireformat::stomp::StompCommandConstants, 3740  
 3740 decaf::net::InetAddress, 2085  
 HEADER\_SELECTOR HOURS  
 activemq::wireformat::stomp::StompCommandConstants, decaf::concurrent::TimeUnit, 3929  
 3740 how  
 HEADER\_SESSIONID gz\_state, 2041

HttpRetryException	activemq::commands::ConnectionControl,
decaf::net::HttpRetryException, 2050,	1306
2051	ID_CONNECTIONERROR
HUFFMAN_ONLY	activemq::commands::ConnectionError,
decaf::util::zip::Deflater, 1769	1336
ID_ACTIVEMQBLOBMESSAGE	ID_CONNECTIONID
activemq::commands::ActiveMQBlobMessage, 1368	activemq::commands::ConnectionId,
189	ID_CONNECTIONINFO
ID_ACTIVEMQBYTESMESSAGE	activemq::commands::ConnectionInfo,
activemq::commands::ActiveMQBytesMessage, 1399	
234	ID_CONSUMERCONTROL
ID_ACTIVEMQDESTINATION	activemq::commands::ConsumerControl,
activemq::commands::ActiveMQDestination, 1446	
323	ID_CONSUMERID
ID_ACTIVEMQMAPMESSAGE	activemq::commands::ConsumerId, 1476
activemq::commands::ActiveMQMapMessage, 1476	ID_CONSUMERINFO
364	activemq::commands::ConsumerInfo,
ID_ACTIVEMQMESSAGE	1509
activemq::commands::ActiveMQMessage, 1509	ID_CONTROLCOMMAND
392	activemq::commands::ControlCommand,
ID_ACTIVEMQOBJECTMESSAGE	1539
activemq::commands::ActiveMQObjectMessage, 1539	ID_DATAARRAYRESPONSE
440	activemq::commands::DataArrayResponse,
ID_ACTIVEMQQUEUE	1574
activemq::commands::ActiveMQQueue, 1574	ID_DATARESPONSE
483	activemq::commands::DataResponse,
ID_ACTIVEMQSTREAMMESSAGE	1634
activemq::commands::ActiveMQStreamMessage, 1634	ID_DESTINATIONINFO
553	activemq::commands::DestinationInfo,
ID_ACTIVEMQTEMPDESTINATION	1784
activemq::commands::ActiveMQTempDestination, 1784	ID_DISCOVERYEVENT
582	activemq::commands::DiscoveryEvent,
ID_ACTIVEMQTEMPQUEUE	1815
activemq::commands::ActiveMQTempQueue, 1815	ID_EXCEPTIONRESPONSE
611	activemq::commands::ExceptionResponse,
ID_ACTIVEMQTEMPTOPIC	1897
activemq::commands::ActiveMQTempTopic, 1897	ID_FLUSHCOMMAND
641	activemq::commands::FlushCommand,
ID_ACTIVEMQTEXTMESSAGE	2001
activemq::commands::ActiveMQTextMessage, 2001	ID_INTEGERRESPONSE
671	activemq::commands::IntegerResponse,
ID_ACTIVEMQTOPIC	2162
activemq::commands::ActiveMQTopic, 2162	ID_JOURNALQUEUEACK
701	activemq::commands::JournalQueueAck,
ID_BROKERID	2227
activemq::commands::BrokerId, 878	ID_JOURNALTOPICACK
ID_BROKERINFO	activemq::commands::JournalTopicAck,
activemq::commands::BrokerInfo, 910	2256
ID_CONNECTIONCONTROL	ID_JOURNALTRACE

activemq::commands::JournalTrace, 2283    activemq::commands::SessionId, 3480  
 ID\_JOURNALTRANSACTION    ID\_SESSIONINFO  
 activemq::commands::JournalTransaction,    activemq::commands::SessionInfo, 3508  
 2311    ID\_SHUTDOWNINFO  
 ID\_KEEPLIVEINFO    activemq::commands::ShutdownInfo,  
 activemq::commands::KeepAliveInfo,    3576  
 2338    ID\_SUBSCRIPTIONINFO  
 ID\_LASTPARTIALCOMMAND    activemq::commands::SubscriptionInfo,  
 activemq::commands::LastPartialCommand,    3786  
 2373    ID\_TRANSACTIONID  
 ID\_LOCALTRANSACTIONID    activemq::commands::TransactionId,  
 activemq::commands::LocalTransactionId,    3935  
 2424    ID\_TRANSACTIONINFO  
 ID\_MESSAGE    activemq::commands::TransactionInfo,  
 activemq::commands::Message, 2613    3964  
 ID\_MESSAGEACK    ID\_WIREFORMATINFO  
 activemq::commands::MessageAck, 2648    activemq::commands::WireFormatInfo,  
 ID\_MESSAGEDISPATCH    4104  
 activemq::commands::MessageDispatch, 2683    ID\_XATransactionId,  
 ID\_MESSAGEDISPATCHNOTIFICATION    4147  
 activemq::commands::MessageDispatchNotification,    4147  
 2720    activemq::util::IdGenerator, 2053  
 ID\_MESSAGEID    IllegalArgumentException  
 activemq::commands::MessageId, 2755    decaf::lang::exceptions::IllegalArgumentException,  
 ID\_MESSAGEPULL    2055, 2056  
 activemq::commands::MessagePull, 2818    IllegalMonitorStateException  
 ID\_NETWORKBRIDGEFILTER    decaf::lang::exceptions::IllegalMonitorStateException,  
 activemq::commands::NetworkBridgeFilter,    2057, 2058  
 2880    IllegalStateException  
 ID\_PARTIALCOMMAND    cms::IllegalStateException, 2060  
 activemq::commands::PartialCommand,    decaf::lang::exceptions::IllegalStateException,  
 3005    2061, 2062  
 ID\_PRODUCERACK    IllegalThreadStateException  
 activemq::commands::ProducerAck, 3128    decaf::lang::exceptions::IllegalThreadStateException,  
 ID\_PRODUCERID    2064, 2065  
 activemq::commands::ProducerId, 3161    impl  
 ID\_PRODUCERINFO    decaf::net::Socket, 3625  
 activemq::commands::ProducerInfo, 3190    implAccept  
 ID\_REMOVEINFO    decaf::net::ServerSocket, 3454  
 activemq::commands::RemoveInfo, 3287  
 ID\_REMOVESEXSCRIPTIONINFO    decaf::io::FileDescriptor, 1949  
 activemq::commands::RemoveSubscriptionInfo, 3318    InactivityMonitor  
 ID\_REPLAYCOMMAND    activemq::transport::inactivity::InactivityMonitor,  
 activemq::commands::ReplayCommand,    2067  
 3347    increaseUsage  
 ID\_RESPONSE    activemq::util::MemoryUsage, 2595  
 activemq::commands::Response, 3382    activemq::util::Usage, 4077  
 ID\_SESSIONID    incrementAndGet

---

decaf::util::concurrent::atomic::AtomicInteger	MEM, 4627
752	NAME, 4627
indexOf	OS, 4626
decaf::util::List, 2412	STORED, 4627
decaf::util::StlList, 3703	SYNC, 4627
IndexOutOfBoundsException	TABLE, 4627
decaf::lang::exceptions::IndexOutOfBoundsException	TYPE, 4627
2070, 2071	TYPEDO, 4627
INDIVIDUAL_ACKNOWLEDGE	
cms::Session, 3464	inflate_mode
Inet4Address	inflate.h, 4626
decaf::net::Inet4Address, 2073	inflate_state, 2085
Inet6Address	back, 2087
decaf::net::Inet6Address, 2077	bits, 2087
InetAddress	check, 2087
decaf::net::Inet4Address, 2076	codes, 2087
decaf::net::Inet6Address, 2077	distbits, 2087
decaf::net::InetAddress, 2080	distcode, 2087
InetSocketAddress	dmax, 2087
decaf::net::InetSocketAddress, 2085	extra, 2087
inffast.h	flags, 2087
OF, 4626	have, 2087
inflate	havedict, 2087
decaf::util::zip::Inflater, 2092, 2093	head, 2087
inflate.h	hold, 2087
BAD, 4627	last, 2087
CHECK, 4627	lenbits, 2087
CODELENS, 4627	lencode, 2087
COMMENT, 4627	length, 2087
COPY, 4627	lens, 2087
COPY_, 4627	mode, 2087
DICT, 4627	ncode, 2087
DICTID, 4627	ndist, 2087
DIST, 4627	next, 2087
DISTEXT, 4627	nlen, 2087
DONE, 4627	offset, 2087
EXLEN, 4626	sane, 2087
EXTRA, 4627	total, 2087
FLAGS, 4626	was, 2087
GUNZIP, 4626	wbits, 2087
HCRC, 4627	whave, 2087
HEAD, 4626	window, 2087
inflate_mode, 4626	wnext, 2087
LEN, 4627	work, 2087
LEN_, 4627	wrap, 2087
LENEXT, 4627	wsize, 2087
LENGTH, 4627	inflateBackInit
LENLENS, 4627	zlib.h, 4635
LIT, 4627	inflateInit
MATCH, 4627	zlib.h, 4635

---

- inflateInit2
  - zlib.h, 4636
- Inflater
  - decaf::util::zip::Inflater, 2090
- inflater
  - decaf::util::zip::InflaterInputStream, 2105
- InflaterInputStream
  - decaf::util::zip::InflaterInputStream, 2100
- INFO
  - decaf::util::logging::Level, 2408
- Info
  - decaf::util::logging, 156
- info
  - decaf::util::logging::Logger, 2470
  - decaf::util::logging::SimpleLogger, 3606
- infrees.h
  - CODES, 4628
  - codetype, 4628
  - DISTS, 4628
  - ENOUGH, 4628
  - ENOUGH\_DISTS, 4628
  - ENOUGH\_LENS, 4628
  - LENS, 4628
  - OF, 4628
- INHERIT
  - decaf::util::logging::Level, 2408
- init
  - activemq::cmsutil::CmsAccessor, 1186
  - activemq::cmsutil::CmsDestinationAccessor, 1189
  - activemq::cmsutil::CmsTemplate, 1207
  - activemq::cmsutil::DestinationResolver, 1811
  - activemq::cmsutil::DynamicDestinationResolver, 1879
- INIT\_STATE
  - deflate.h, 4623
- initCause
  - decaf::lang::Exception, 1891
  - decaf::lang::Throwable, 3898
- initializeLibrary
  - activemq::library::ActiveMQCPP, 311
- initializeNetworking
  - decaf::internal::net::Network, 2877
- initializeRuntime
  - decaf::lang::Runtime, 3420
- initSocketImpl
  - decaf::net::Socket, 3620
- inProgressClearRequired
  - activemq::core::ActiveMQConsumer, 307
- InputStream
  - decaf::io::InputStream, 2107
- inputStream
  - decaf::io::FilterInputStream, 1957
- InputStreamReader
  - decaf::io::InputStreamReader, 2117
- inReceive
  - activemq::wireformat::openwire::OpenWireFormat, 2977
  - activemq::wireformat::stomp::StompWireFormat, 3753
  - activemq::wireformat::WireFormat, 4090
- jas\_h
  - internal\_state, 2191
- insert
  - decaf::internal::util::TimerTaskHeap, 3918
- IntArrayBuffer
  - decaf::internal::nio::IntArrayBuffer, 2123, 2124
- intBitsToFloat
  - decaf::lang::Float, 1967
- IntBuffer
  - decaf::nio::IntBuffer, 2133
- Integer
  - decaf::lang::Integer, 2146
- INTEGER\_TYPE
  - activemq::util::PrimitiveValueNode, 3104
- IntegerResponse
  - activemq::commands::IntegerResponse, 2161
- IntegerResponseMarshaller
  - activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller, 2181
  - activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller, 2168
  - activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, 2172
  - activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller, 2177
  - activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller, 2185
  - activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller, 2164
- internal\_state, 2188
  - bi\_buf, 2191
  - bi\_valid, 2191
  - bl\_count, 2191

- bl\_desc, 2191
- bl\_tree, 2191
- block\_start, 2191
- d\_buf, 2191
- d\_desc, 2191
- depth, 2191
- dummy, 2191
- dyn\_dtree, 2191
- dyn\_ltree, 2191
- good\_match, 2191
- gzhead, 2191
- gzindex, 2191
- hash\_bits, 2191
- hash\_mask, 2191
- hash\_shift, 2191
- hash\_size, 2191
- head, 2191
- heap, 2191
- heap\_len, 2191
- heap\_max, 2191
- high\_water, 2191
- ins\_h, 2191
- l\_buf, 2191
- l\_desc, 2191
- last\_eob\_len, 2191
- last\_flush, 2191
- last\_lit, 2191
- level, 2191
- lit\_bufsize, 2191
- lookahead, 2191
- match\_available, 2191
- match\_length, 2191
- match\_start, 2191
- matches, 2191
- max\_chain\_length, 2191
- max\_lazy\_match, 2191
- method, 2191
- nice\_match, 2191
- opt\_len, 2191
- pending, 2191
- pending\_buf, 2191
- pending\_buf\_size, 2191
- pending\_out, 2191
- prev, 2191
- prev\_length, 2191
- prev\_match, 2191
- static\_len, 2191
- status, 2191
- strategy, 2191
- strm, 2191
- strstart, 2191
- w\_bits, 2191
- w\_mask, 2191
- w\_size, 2191
- window, 2191
- window\_size, 2191
- wrap, 2191
- InternalCommandListener
  - activemq::transport::mock::InternalCommandListener, 2193
- InterruptedException
  - decaf::lang::exceptions::InterruptedException, 2194, 2195
- InterruptedIOException
  - decaf::io::InterruptedIOException, 2197, 2198
- intf
  - zconf.h, 4632
- intValue
  - activemq::util::PrimitiveValueNode::PrimitiveValue, 3098
  - decaf::lang::Byte, 974
  - decaf::lang::Character, 1131
  - decaf::lang::Double, 1847
  - decaf::lang::Float, 1968
  - decaf::lang::Integer, 2150
  - decaf::lang::Long, 2501
  - decaf::lang::Number, 2920
  - decaf::lang::Short, 3543
  - decaf::util::concurrent::atomic::AtomicInteger, 752
  - decaf::util::logging::Level, 2406
- InvalidClientIdException
  - cms::InvalidClientIdException, 2200
- InvalidDestinationException
  - cms::InvalidDestinationException, 2201
- InvalidKeyException
  - decaf::security::InvalidKeyException, 2202, 2203
- InvalidMarkException
  - decaf::nio::InvalidMarkException, 2204, 2205
- InvalidSelectorException
  - cms::InvalidSelectorException, 2207
- InvalidStateException
  - decaf::lang::exceptions::InvalidStateException, 2208, 2209
- IOException
  - decaf::io::IOException, 2211, 2212
- IOTransport



- activemq::transport::IOTransport, 2215
- isClientMaster
  - activemq::commands::ConnectionInfo, 1397
- IPos
  - deflate.h, 4623
- isAbsolute
  - decaf::internal::net::URIType, 4068
  - decaf::net::URI, 4039
- isClose
  - activemq::commands::ConnectionControl, 1305
- isAdvisory
  - activemq::commands::ConsumerControl, 1445
- activemq::commands::ActiveMQDestination, 319
- isClosed
  - activemq::core::ActiveMQConnection, 273
- isAlive
  - decaf::lang::Thread, 3882
- isAlwaysSyncSend
  - activemq::core::ActiveMQConnection, 273
  - activemq::core::ActiveMQConnectionFactory, 289
  - activemq::core::ActiveMQConsumer, 307
  - activemq::core::ActiveMQProducer, 470
  - activemq::core::MessageDispatchChannel, 2686
- isAnyLocalAddress
  - decaf::net::Inet4Address, 2073
  - decaf::net::Inet6Address, 2082
- activemq::transport::failover::BackupTransport, 760
- isAutoAcknowledge
  - activemq::core::ActiveMQSession, 527
- activemq::transport::failover::FailoverTransport, 1935
- isBackup
  - activemq::transport::mock::MockTransport, 2859
- activemq::transport::tcp::TcpTransport, 3868
- isBound
  - decaf::net::ServerSocket, 3454
  - decaf::net::Socket, 3620
- activemq::transport::Transport, 3998
- activemq::transport::TransportFilter, 4008
- isBrokerInfo
  - decaf::internal::net::tcp::TcpSocket, 3857
  - decaf::io::FilterInputStream, 1955
  - decaf::io::FilterOutputStream, 1960
  - decaf::net::ServerSocket, 3455
  - decaf::net::Socket, 3620
- activemq::commands::BaseCommand, 767
- activemq::commands::BrokerInfo, 907
- activemq::commands::Command, 1228
- isBrokerMasterConnector
  - activemq::commands::ConnectionInfo, 1397
- isComposite
  - activemq::commands::ActiveMQDestination, 319
- isBrowser
  - activemq::commands::ConsumerInfo, 1506
- isCompressed
  - activemq::commands::Message, 2607
- isConnected
  - activemq::transport::failover::FailoverTransport, 1935
  - activemq::transport::IOTransport, 2216
  - activemq::transport::mock::MockTransport, 2859
  - activemq::transport::tcp::TcpTransport, 3868
  - activemq::transport::Transport, 3998
  - activemq::transport::TransportFilter, 4008
- isBusy
  - decaf::util::concurrent::PooledThread, 3057
- isCacheEnabled
  - activemq::commands::WireFormatInfo, 4099
  - activemq::wireformat::openwire::OpenWireFormat, 2977
- decaf::internal::net::tcp::TcpSocket, 3857
- isCancelled
  - decaf::util::concurrent::Future, 2032
- decaf::net::Socket, 3620
- isClientAcknowledge
  - activemq::core::ActiveMQSession, 527
- isConnectionAdvisory

- activemq::commands::ActiveMQDestination, 319
- isConnectionInfo
  - activemq::commands::BaseCommand, 768
  - activemq::commands::Command, 1228
  - activemq::commands::ConnectionInfo, 1397
- isConnectionInterruptProcessingComplete
  - activemq::state::ConnectionState, 1432
- isConsumerAdvisory
  - activemq::commands::ActiveMQDestination, 319
- isConsumerInfo
  - activemq::commands::BaseCommand, 768
  - activemq::commands::Command, 1229
  - activemq::commands::ConsumerInfo, 1506
- isDeletedByBroker
  - activemq::commands::ActiveMQBlobMessage, 188
- isDigit
  - decaf::lang::Character, 1131
- isDispatchAsync
  - activemq::commands::ConsumerInfo, 1506
  - activemq::commands::ProducerInfo, 3188
  - activemq::core::ActiveMQConnection, 273
  - activemq::core::ActiveMQConnectionFactory, 289
- isDone
  - decaf::util::concurrent::Future, 2032
  - decaf::util::zip::DeflaterOutputStream, 1774
- isDroppable
  - activemq::commands::Message, 2607
- isDuplexConnection
  - activemq::commands::BrokerInfo, 907
- isDupsOkAcknowledge
  - activemq::core::ActiveMQSession, 527
- isEmpty
  - activemq::core::ActiveMQSessionExecutor, 534
  - activemq::core::MessageDispatchChannel, 2686
  - activemq::util::ActiveMQProperties, 478
  - cms::CMSProperties, 1198
- decaf::internal::util::TimerTaskHeap, 3918
- decaf::lang::String, 3777
- decaf::util::AbstractCollection, 166
- decaf::util::Collection, 1223
- decaf::util::concurrent::ConcurrentStlMap, 1273
- decaf::util::concurrent::SynchronousQueue, 3833
- decaf::util::Map, 2545
- decaf::util::Properties, 3220
- decaf::util::StlList, 3704
- decaf::util::StlMap, 3716
- decaf::util::StlSet, 3736
- isEnabled
  - activemq::transport::failover::BackupTransportPool, 763
- isExclusive
  - activemq::commands::ActiveMQDestination, 319
  - activemq::commands::ConsumerInfo, 1507
- isExit
  - activemq::commands::ConnectionControl, 1305
- isExpired
  - activemq::commands::Message, 2607
- isExplicitQosEnabled
  - activemq::cmsutil::CmsTemplate, 1207
- isFailOnClose
  - activemq::transport::mock::MockTransport, 2859
- isFailOnKeepAliveSends
  - activemq::transport::mock::MockTransport, 2859
- isFailOnReceiveMessage
  - activemq::transport::mock::MockTransport, 2859
- isFailOnSendMessage
  - activemq::transport::mock::MockTransport, 2859
- isFailOnStart
  - activemq::transport::mock::MockTransport, 2859
- isFailOnStop
  - activemq::transport::mock::MockTransport, 2859
- isFair
  - decaf::util::concurrent::locks::ReentrantLock, 3275

- decaf::util::concurrent::Semaphore, 344
- isFaultTolerant
  - activemq::commands::ConnectionControlLetterOrDigit, 1305
  - activemq::commands::ConnectionInfo, isLinkLocalAddress, 1397
  - activemq::transport::failover::FailoverTransport, isLocked, 1936
  - activemq::transport::IOTransport, 2216
  - activemq::transport::mock::MockTransport, decaf::util::concurrent::locks::ReentrantLock, 2859
  - activemq::transport::tcp::TcpTransport, isLoggable, 3868
  - activemq::transport::Transport, 3999
  - activemq::transport::TransportFilter, 4009
- isFaultTolerantConfiguration
  - activemq::commands::BrokerInfo, 908
- isFlush
  - activemq::commands::ConsumerControl, 1445
- isFull
  - activemq::util::MemoryUsage, 2595
  - activemq::util::Usage, 4077
- isHeldByCurrentThread
  - decaf::util::concurrent::locks::ReentrantLock, 3275
- isIndividualAcknowledge
  - activemq::core::ActiveMQSession, 527
- isInfinite
  - decaf::lang::Double, 1847
  - decaf::lang::Float, 1968
- isInitialized
  - activemq::transport::failover::FailoverTransport, 1936
- isInputShutdown
  - decaf::net::Socket, 3620
- isInTransaction
  - activemq::core::ActiveMQTransactionContext, 729
- isISOCtrl
  - decaf::lang::Character, 1131
- isKeepAliveInfo
  - activemq::commands::BaseCommand, 768
  - activemq::commands::Command, 1229
  - activemq::commands::KeepAliveInfo, 2337
- isKeepAliveResponseRequired
  - activemq::transport::inactivity::InactivityMonitor, 2067
- isLetter
  - decaf::lang::Character, 1131
- isLetterOrDigit
  - decaf::lang::Character, 1131
- isLinkLocalAddress
  - decaf::net::Inet4Address, 2074
- isNetAddress
  - decaf::net::InetAddress, 2082
- isLocked
  - decaf::util::concurrent::Lock, 2451
- isLoggable
  - decaf::util::logging::Filter, 1950
  - decaf::util::logging::Handler, 2044
  - decaf::util::logging::Logger, 2470
  - decaf::util::logging::StreamHandler, 3759
- isLoopbackAddress
  - decaf::net::Inet4Address, 2074
- isLowerCase
  - decaf::lang::Character, 1131
- isManageable
  - activemq::commands::ConnectionInfo, 1398
- isMarshalAware
  - activemq::commands::ActiveMQMapMessage, 359
  - activemq::commands::BaseDataStructure, 840
  - activemq::commands::Message, 2607
  - activemq::commands::WireFormatInfo, 4099
- isNotWireFormatAware
  - activemq::wireformat::MarshalAware, 2566
- isMasterBroker
  - activemq::commands::BrokerInfo, 908
- isMCGlobal
  - decaf::net::Inet4Address, 2074
  - decaf::net::InetAddress, 2083
- isMCLinkLocal
  - decaf::net::Inet4Address, 2074
- isMCNodeLocal
  - decaf::net::Inet4Address, 2074
- isMCOrgLocal
  - decaf::net::Inet4Address, 2075
- isMCSiteLocal
  - decaf::net::InetAddress, 2083
- isMCSubnetLocal
  - decaf::net::Inet4Address, 2075

- decaf::net::InetAddress, 2084
- isMessage
  - activemq::commands::BaseCommand, isOrdered 768
  - activemq::commands::Command, 1229
  - activemq::commands::Message, 2608
- isMessageAck
  - activemq::commands::BaseCommand, isPending 768
  - activemq::commands::Command, 1229
  - activemq::commands::MessageAck, 2646
- isMessageDispatch
  - activemq::commands::BaseCommand, 768
  - activemq::commands::Command, 1229
  - activemq::commands::MessageDispatch, 2681
- isMessageDispatchNotification
  - activemq::commands::BaseCommand, 768
  - activemq::commands::Command, 1229
  - activemq::commands::MessageDispatchNotification, 2719
- isMessageIdEnabled
  - activemq::cmsutil::CmsTemplate, 1208
- isMessageTimestampEnabled
  - activemq::cmsutil::CmsTemplate, 1208
- isMulticastAddress
  - decaf::net::Inet4Address, 2075
  - decaf::net::InetAddress, 2084
- isNaN
  - decaf::lang::Double, 1848
  - decaf::lang::Float, 1968
- isNetworkConnection
  - activemq::commands::BrokerInfo, 908
- isNetworkSubscription
  - activemq::commands::ConsumerInfo, 1507
- isNoLocal
  - activemq::cmsutil::CmsTemplate, 1208
  - activemq::commands::ConsumerInfo, 1507
- isNoRangeAcks
  - activemq::commands::ConsumerInfo, 1507
- isOpaque
  - decaf::internal::net::URIType, 4068
  - decaf::net::URI, 4040
- isOptimizedAcknowledge
  - activemq::commands::ConsumerInfo, 1507
  - activemq::commands::ActiveMQDestination, 319
  - decaf::net::Socket, 3621
  - activemq::threads::CompositeTask, 1255
  - activemq::transport::failover::BackupTransportPool, 763
  - activemq::transport::failover::CloseTransportsTask, 1182
  - activemq::transport::failover::FailoverTransport, 1936
  - isPersistent
    - activemq::commands::Message, 2608
  - isPrepared
    - activemq::state::TransactionState, 3991
  - isProducerAck
    - activemq::commands::BaseCommand, 769
    - activemq::commands::Command, 1229
    - activemq::commands::ProducerAck, 3127
  - isProducerAdvisory
    - activemq::commands::ActiveMQDestination, 319
  - isProducerInfo
    - activemq::commands::BaseCommand, 769
    - activemq::commands::Command, 1230
    - activemq::commands::ProducerInfo, 3188
  - isPubSubDomain
    - activemq::cmsutil::CmsDestinationAccessor, 1189
  - isQueue
    - activemq::commands::ActiveMQDestination, 320
  - isRandomize
    - activemq::transport::failover::FailoverTransport, 1936
    - activemq::transport::failover::URIPool, 4056
  - isReadOnly
    - decaf::internal::nio::ByteBuffer, 1030
    - decaf::internal::nio::CharArrayBuffer, 1145
    - decaf::internal::nio::DoubleArrayBuffer, 1863

- decaf::internal::nio::FloatArrayBuffer, isResume
  - 1983
  - activemq::commands::ConnectionControl, 1305
- decaf::internal::nio::IntArrayBuffer, 2128
- decaf::internal::nio::LongArrayBuffer, isRetroactive
  - 2520
  - activemq::commands::ConsumerInfo, 1507
- decaf::internal::nio::ShortArrayBuffer,
  - 3558
  - isRunning
- decaf::nio::Buffer, 940
- decaf::nio::ByteBuffer, 1068
- isReadOnlyBody
  - activemq::commands::Message, 2608
  - activemq::core::MessageDispatchChannel, 2686
- isReadOnlyProperties
  - activemq::commands::Message, 2608
  - isScheduled
  - decaf::util::TimerTask, 3916
- isRebalanceConnection
  - activemq::commands::ConnectionControl, 1305
  - isServerAuthority
  - decaf::internal::net::URIType, 4068
  - isShutdownInfo
- isRecievedByDFBridge
  - activemq::commands::Message, 2608
  - activemq::commands::BaseCommand, 769
  - activemq::commands::Command, 1230
  - activemq::commands::RemoveInfo, 3286
- isRemoveInfo
  - activemq::commands::BaseCommand, 769
  - activemq::commands::ShutdownInfo, 3575
  - activemq::commands::Command, 1230
  - isSiteLocalAddress
  - decaf::net::Inet4Address, 2075
  - decaf::net::InetAddress, 2084
- isRemoveSubscriptionInfo
  - activemq::commands::BaseCommand, 769
  - isSizePrefixDisabled
  - activemq::commands::WireFormatInfo, 4100
  - activemq::wireformat::openwire::OpenWireFormat, 2977
- isResponse
  - activemq::commands::BaseCommand, 769
  - activemq::commands::Command, 1230
  - activemq::commands::Response, 3381
  - isSlaveBroker
  - activemq::commands::BrokerInfo, 908
  - isStackTraceEnabled
  - activemq::commands::WireFormatInfo, 4100
- isResponseRequired
  - activemq::commands::BaseCommand, 769
  - activemq::wireformat::openwire::OpenWireFormat, 2978
  - isStart
  - activemq::commands::Command, 1230
  - activemq::commands::ConsumerControl, 1445
- isRestoreConsumers
  - activemq::state::ConnectionStateTracker, 1435
  - isStarted
  - activemq::core::ActiveMQConnection, 273
- isRestoreProducers
  - activemq::state::ConnectionStateTracker, 1435
  - activemq::core::ActiveMQSession, 528
- isRestoreSessions
  - activemq::state::ConnectionStateTracker, 1435
  - isStop
  - activemq::commands::ConsumerControl, 1445
- isRestoreTransaction
  - activemq::state::ConnectionStateTracker, 1435
  - isSuspend
  - activemq::commands::ConnectionControl, 1305
  - isSynchronizationRegistered

activemq::core::ActiveMQConsumer,	activemq::core::ActiveMQConnectionFactory,
307	289
isTcpNoDelayEnabled	isUseCollisionAvoidance
activemq::commands::WireFormatInfo,	activemq::core::policies::DefaultRedeliveryPolicy,
4100	1733
activemq::wireformat::openwire::OpenWireFormat,	activemq::core::RedeliveryPolicy, 3271
2978	isUseCompression
isTemporary	activemq::core::ActiveMQConnection,
activemq::commands::ActiveMQDestination,	274
320	activemq::core::ActiveMQConnectionFactory,
isTightEncodingEnabled	289
activemq::commands::WireFormatInfo,	isUseExponentialBackOff
4100	activemq::core::policies::DefaultRedeliveryPolicy,
activemq::wireformat::openwire::OpenWireFormat,	1733
2978	activemq::core::RedeliveryPolicy, 3271
isTopic	activemq::transport::failover::FailoverTransport,
activemq::commands::ActiveMQDestination,	1937
320	isValid
isTrackMessages	activemq::commands::WireFormatInfo,
activemq::state::ConnectionStateTracker,	4100
1435	decaf::internal::net::URIType, 4068
activemq::transport::failover::FailoverTransport,	isValidDomainName
1937	decaf::internal::net::URIHelper, 4049
isTrackTransactionProducers	isValidHexChar
activemq::state::ConnectionStateTracker,	decaf::internal::net::URIHelper, 4049
1435	isValidHost
activemq::transport::failover::FailoverTransport,	decaf::internal::net::URIHelper, 4049
1937	isValidIP4Word
isTrackTransactions	decaf::internal::net::URIHelper, 4050
activemq::state::ConnectionStateTracker,	isValidIP6Address
1435	decaf::internal::net::URIHelper, 4050
isTransacted	isValidIPv4Address
activemq::cmsutil::PooledSession, 3054	decaf::internal::net::URIHelper, 4050
activemq::core::ActiveMQSession, 528	isWaitingForResponse
cms::Session, 3473	activemq::state::Tracked, 3932
isTransactionInfo	isWhitespace
activemq::commands::BaseCommand,	decaf::lang::Character, 1132
770	isWildcard
activemq::commands::Command, 1230	activemq::commands::ActiveMQDestination,
activemq::commands::TransactionInfo,	320
3962	isWireFormatInfo
isTransportFailed	activemq::commands::BaseCommand,
activemq::core::ActiveMQConnection,	770
273	activemq::commands::Command, 1231
isUpperCase	activemq::commands::WireFormatInfo,
decaf::lang::Character, 1132	4101
isUseAsyncSend	itemExists
activemq::core::ActiveMQConnection,	activemq::commands::ActiveMQMapMessage,
274	360
	cms::MapMessage, 2558

- iterate
  - activemq::core::ActiveMQConsumer, 308
  - activemq::core::ActiveMQSessionExecutor, 535
  - activemq::threads::CompositeTaskRunner, 1257
  - activemq::threads::Task, 3847
  - activemq::transport::failover::BackupTransportPool, 764
  - activemq::transport::failover::CloseTransportTask, 1182
  - activemq::transport::failover::FailoverTransport, 1937
- iterator
  - decaf::lang::Iterable, 2221, 2222
  - decaf::util::concurrent::SynchronousQueue, 3833
  - decaf::util::PriorityQueue, 3121
  - decaf::util::StlList, 3704
  - decaf::util::StlQueue, 3726
  - decaf::util::StlSet, 3736
- join
  - decaf::lang::Thread, 3883
- JournalQueueAck
  - activemq::commands::JournalQueueAck, 2225
- JournalQueueAckMarshaller
  - activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller, 2249
  - activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller, 2233
  - activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller, 2241
  - activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller, 2245
  - activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller, 2237
  - activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller, 2229
- JournalTopicAck
  - activemq::commands::JournalTopicAck, 2253
- JournalTopicAckMarshaller
  - activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller, 2278
  - activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller, 2262
- activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller, 2266
- activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller, 2274
- activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller, 2258
- activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller, 2270
- activemq::commands::JournalTrace, 2282
- activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller, 2301
- activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller, 2285
- activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller, 2289
- activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller, 2297
- activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller, 2305
- activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller, 2293
- JournalTransaction
  - activemq::commands::JournalTransaction, 2309
- JournalTransactionMarshaller
  - activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller, 2332
  - activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller, 2316
  - activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller, 2320
  - activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller, 2328
  - activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller, 2324
  - activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller, 2312
- KeepAliveInfo
  - activemq::commands::KeepAliveInfo, 2336
- KeepAliveInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller, 2361
  - activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller, 2344
  - activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller, 2348

activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller, 2352  
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller, 2356  
 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller, 2339  
 KeyException, 2388  
 KeyException, activemq::wireformat::openwire::marshal::v6::LastPartialCommand, 2375  
 decaf::security::KeyException, 2367, 2368  
 KeyManagementException, inflate.h, 4627  
 decaf::security::KeyManagementException, 2369, 2370  
 deflate.h, 4623  
 keySet, len  
 decaf::util::concurrent::ConcurrentStlMap, ct\_data\_s, 1571  
 1273  
 decaf::util::Map, 2546  
 decaf::util::StlMap, 3716  
 lenbits  
 inflate\_state, 2087  
 l\_buf, lencode  
 internal\_state, 2191  
 L\_CODES, inflate\_state, 2087  
 deflate.h, 4623  
 LENEXT  
 inflate.h, 4627  
 l\_desc, LENGTH  
 internal\_state, 2191  
 inflate.h, 4627  
 last, length  
 inflate\_state, 2087  
 last\_eob\_len, decaf::internal::nio::CharArrayBuffer, 1148  
 internal\_state, 2191  
 last\_flush, decaf::lang::ArrayPointer, 741  
 internal\_state, 2191  
 last\_lit, decaf::lang::CharSequence, 1168  
 internal\_state, 2191  
 lastMessageId, decaf::lang::String, 3778  
 internal\_state, 2191  
 lastDeliveredSequenceId, decaf::nio::CharBuffer, 1159  
 activemq::commands::RemoveInfo, 3287  
 lastIndexOf, decaf::util::zip::InflaterInputStream, 2105  
 decaf::util::List, 2413  
 decaf::util::StlList, 3704  
 lastMessageId, inflate\_state, 2087  
 activemq::commands::MessageAck, 2648  
 lastNakNumber, LENGTH\_CODES  
 activemq::commands::ReplayCommand, 3347  
 Less  
 LastPartialCommand, deflate.h, 4623  
 activemq::commands::LastPartialCommand, 2372  
 LastPartialCommandMarshaller, inflate.h, 4627  
 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller, 2396  
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller, 2383  
 decaf::util::comparators::Less, 2400  
 decaf::util::logging::Level, 2405  
 level, 2404  
 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller, 2396  
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller, 2383  
 decaf::util::logging, 155



- limit
  - decaf::nio::Buffer, 940
- LineNumber
  - activemq::commands::BrokerError::StackTraceElement, 2687
  - 3686
- List
  - decaf::util::List, 2410
- LIST\_TYPE
  - activemq::util::PrimitiveValueNode, 3104
- listen
  - decaf::internal::net::tcp::TcpSocket, 3857
  - decaf::net::SocketImpl, 3642
- listener
  - activemq::transport::TransportFilter, 4013
- listIterator
  - decaf::util::List, 2413, 2414
  - decaf::util::StlList, 3705, 3706
- listValue
  - activemq::util::PrimitiveValueNode::PrimitiveValue, 3098
- LIT
  - inflate.h, 4627
- lit\_bufsize
  - internal\_state, 2191
- LITERALS
  - deflate.h, 4623
- load
  - decaf::util::Properties, 3221
- local
  - gzguts.h, 4625
  - zutil.h, 4639
- localPort
  - decaf::net::SocketImpl, 3644
- LocalTransactionId
  - activemq::commands::LocalTransactionId, 2422
- LocalTransactionIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller, 2447
  - activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller, 2430
  - activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller, 2434
  - activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller, 2443
  - activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller, 2439
  - activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller, 2426
- Lock
  - decaf::util::concurrent::Lock, 2451
- lock
  - activemq::core::MessageDispatchChannel, 2687
  - decaf::internal::util::concurrent::MutexImpl, 2873
  - decaf::internal::util::concurrent::SynchronizableImpl, 3823
  - decaf::io::InputStream, 2109
  - decaf::io::OutputStream, 2994
  - decaf::util::AbstractCollection, 166
  - decaf::util::concurrent::ConcurrentStlMap, 1274
  - decaf::util::concurrent::Lock, 2451
  - decaf::util::concurrent::locks::Lock, 2454
  - decaf::util::concurrent::locks::ReentrantLock, 3276
  - decaf::util::concurrent::Mutex, 2868
  - decaf::util::concurrent::Synchronizable, 3812
  - decaf::util::StlMap, 3717
  - decaf::util::StlQueue, 3726
- lock\_count
  - decaf::util::concurrent::MutexHandle, 2872
- lock\_owner
  - decaf::util::concurrent::MutexHandle, 2872
- lockInterruptibly
  - decaf::util::concurrent::locks::Lock, 2454
  - decaf::util::concurrent::locks::ReentrantLock, 3276
- log
  - decaf::util::logging::Logger, 2470, 2471
  - decaf::util::logging::LogWriter, 2494
  - decaf::util::logging::SimpleLogger, 3607
- LOGDECAF\_DEBUG
  - LoggerDefines.h, 4736
- LOGDECAF\_DEBUG\_1
  - LoggerDefines.h, 4736
- LOGDECAF\_DECLARE
  - LoggerDefines.h, 4736
- LOGDECAF\_DECLARE\_LOCAL
  - LoggerDefines.h, 4736
- LOGDECAF\_ERROR
  - LoggerDefines.h, 4736
- LOGDECAF\_FATAL
  - LoggerDefines.h, 4736
- LOGDECAF\_INFO
  - LoggerDefines.h, 4737

- LOGDECAF\_INITIALIZE
  - LoggerDefines.h, 4737
- LOGDECAF\_WARN
  - LoggerDefines.h, 4737
- Logger
  - decaf::util::logging::Logger, 2465
- LoggerDefines.h
  - LOGDECAF\_DEBUG, 4736
  - LOGDECAF\_DEBUG\_1, 4736
  - LOGDECAF\_DECLARE, 4736
  - LOGDECAF\_DECLARE\_LOCAL, 4737
  - LOGDECAF\_ERROR, 4737
  - LOGDECAF\_FATAL, 4737
  - LOGDECAF\_INFO, 4737
  - LOGDECAF\_INITIALIZE, 4737
  - LOGDECAF\_WARN, 4737
- LoggerHierarchy
  - decaf::util::logging::LoggerHierarchy, 2474
- LoggingInputStream
  - activemq::io::LoggingInputStream, 2475
- LoggingOutputStream
  - activemq::io::LoggingOutputStream, 2476
- LoggingTransport
  - activemq::transport::logging::LoggingTransport, 2478
- LogManager
  - decaf::util::logging::LogManager, 2483
- LogRecord
  - decaf::util::logging::LogRecord, 2489
- LogWriter
  - decaf::util::logging::LogWriter, 2494
- Long
  - decaf::lang::Long, 2498
- LONG\_TYPE
  - activemq::util::PrimitiveValueNode, 3104
- LongArrayBuffer
  - decaf::internal::nio::LongArrayBuffer, 2515, 2516
- longBitsToDouble
  - decaf::lang::Double, 1848
- LongBuffer
  - decaf::nio::LongBuffer, 2525
- LongSequenceGenerator
  - activemq::util::LongSequenceGenerator, 2535
- longValue
  - activemq::util::PrimitiveValueNode::PrimitiveValueNode, 3098
- decaf::lang::Byte, 974
- decaf::lang::Character, 1132
- decaf::lang::Double, 1848
- decaf::lang::Float, 1969
- decaf::lang::Integer, 2150
- decaf::lang::Long, 2501
- decaf::lang::Number, 2920
- decaf::lang::Short, 3543
- decaf::util::concurrent::atomic::AtomicInteger, 753
- LOOK
  - gzguts.h, 4625
- lookahead
  - internal\_state, 2191
- loopbackBytes
  - decaf::net::InetAddress, 2085
- looseMarshal
  - activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 820
  - activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1675
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBlob, 195
  - activemq::wireformat::openwire::marshal::v1::ActiveMQByte, 241
  - activemq::wireformat::openwire::marshal::v1::ActiveMQDestination, 329
  - activemq::wireformat::openwire::marshal::v1::ActiveMQMap, 370
  - activemq::wireformat::openwire::marshal::v1::ActiveMQMessage, 399
  - activemq::wireformat::openwire::marshal::v1::ActiveMQObject, 446
  - activemq::wireformat::openwire::marshal::v1::ActiveMQQueue, 493
  - activemq::wireformat::openwire::marshal::v1::ActiveMQStream, 559
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTemp, 588
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTemp, 617
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTemp, 651
  - activemq::wireformat::openwire::marshal::v1::ActiveMQText, 682
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTopic, 711
  - activemq::wireformat::openwire::marshal::v1::BaseCommand, 787

activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller	888	activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller	2709
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller	921	activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller	2739
activemq::wireformat::openwire::marshal::v1::ConnectiveControlMarshaller	1317	activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller	2777
activemq::wireformat::openwire::marshal::v1::ConnectiveFromMarshaller	1350	activemq::wireformat::openwire::marshal::v1::MessageMarshaller	2799
activemq::wireformat::openwire::marshal::v1::ConnectiveToMarshaller	1382	activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller	2847
activemq::wireformat::openwire::marshal::v1::ConnectiveToMarshaller	1414	activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller	2902
activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller	1461	activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	3030
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller	1491	activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller	3151
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	1524	activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller	3183
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	1554	activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	3200
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	1589	activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller	3302
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	1658	activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller	3320
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	1799	activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller	3353
activemq::wireformat::openwire::marshal::v1::DiscoveryInfoMarshaller	1833	activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	3410
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	1921	activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller	3502
activemq::wireformat::openwire::marshal::v1::FlushWireFormatMarshaller	2020	activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller	3519
activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller	2181	activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller	3586
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	2249	activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	3792
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller	2278	activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller	3941
activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller	2301	activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller	3970
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller	2333	activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller	4122
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	2361	activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller	4161
activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller	2397	activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller	204
activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller	2448	activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller	258
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller	2667	activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller	341

activemq::wireformat::openwire::marshal::v2::ActiveMQInfoMessageMarshaller, marshal::v2::ExceptionResponse 383 1904  
 activemq::wireformat::openwire::marshal::v2::ActiveMQInfoMessageMarshaller, marshal::v2::FlushCommand 411 2008  
 activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller, marshal::v2::IntegerResponse 459 2168  
 activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller, marshal::v2::JournalQueueAck 505 2233  
 activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMessageMarshaller, marshal::v2::JournalTopicAck 572 2262  
 activemq::wireformat::openwire::marshal::v2::ActiveMQReplyDestinationMarshaller, marshal::v2::JournalTraceMessage 600 2285  
 activemq::wireformat::openwire::marshal::v2::ActiveMQReplyQueueMarshaller, marshal::v2::JournalTransaction 630 2317  
 activemq::wireformat::openwire::marshal::v2::ActiveMQReplyTopicMarshaller, marshal::v2::KeepAliveInfo 660 2344  
 activemq::wireformat::openwire::marshal::v2::ActiveMQReplyMessageMarshaller, marshal::v2::LastPartialCommand 694 2384  
 activemq::wireformat::openwire::marshal::v2::ActiveMQReplyMessageMarshaller, marshal::v2::LocalTransaction 724 2431  
 activemq::wireformat::openwire::marshal::v2::ActiveMQReplyMessageMarshaller, marshal::v2::MessageAckMessage 808 2654  
 activemq::wireformat::openwire::marshal::v2::ActiveMQReplyMessageMarshaller, marshal::v2::MessageDispatch 900 2692  
 activemq::wireformat::openwire::marshal::v2::ActiveMQReplyMessageMarshaller, marshal::v2::MessageDispatch 933 2726  
 activemq::wireformat::openwire::marshal::v2::ActiveMQReplyMessageMarshaller, marshal::v2::MessageIdMarshaller 1330 2757  
 activemq::wireformat::openwire::marshal::v2::ActiveMQReplyMessageMarshaller, marshal::v2::MessageMarshaller 1338 2790  
 activemq::wireformat::openwire::marshal::v2::ActiveMQReplyMessageMarshaller, marshal::v2::MessagePullMarshaller 1370 2830  
 activemq::wireformat::openwire::marshal::v2::ActiveMQReplyMessageMarshaller, marshal::v2::NetworkBridge 1401 2882  
 activemq::wireformat::openwire::marshal::v2::ActiveMQReplyMessageMarshaller, marshal::v2::PartialCommand 1448 3012  
 activemq::wireformat::openwire::marshal::v2::ActiveMQReplyMessageMarshaller, marshal::v2::ProducerAckMessage 1479 3130  
 activemq::wireformat::openwire::marshal::v2::ActiveMQReplyMessageMarshaller, marshal::v2::ProducerIdMarshaller 1511 3163  
 activemq::wireformat::openwire::marshal::v2::ActiveMQReplyMessageMarshaller, marshal::v2::ProducerInfoMarshaller 1541 3196  
 activemq::wireformat::openwire::marshal::v2::ActiveMQReplyMessageMarshaller, marshal::v2::RemoveInfoMarshaller 1576 3289  
 activemq::wireformat::openwire::marshal::v2::ActiveMQReplyMessageMarshaller, marshal::v2::RemoveSubscription 1645 3328  
 activemq::wireformat::openwire::marshal::v2::ActiveMQReplyMessageMarshaller, marshal::v2::ReplayCommand 1786 3357  
 activemq::wireformat::openwire::marshal::v2::ActiveMQReplyMessageMarshaller, marshal::v2::ResponseMarshaller 1821 3395

activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller,	1342
3482	
activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller,	1374
3527	
activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller,	1406
3582	
activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller,	1453
3808	
activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller,	1483
3945	
activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller,	1516
3987	
activemq::wireformat::openwire::marshal::v2::WireFormatMarshaller,	1545
4114	
activemq::wireformat::openwire::marshal::v2::XaTransactionIdMarshaller,	1580
4153	
activemq::wireformat::openwire::marshal::v3::ActiveMQDeflatingMessageMarshaller,	1649
191	
activemq::wireformat::openwire::marshal::v3::ActiveMQDeflatingMessageMarshaller,	1790
236	
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller,	1825
325	
activemq::wireformat::openwire::marshal::v3::ActiveMQFormatMessageMarshaller,	1908
366	
activemq::wireformat::openwire::marshal::v3::ActiveMQFormatMessageMarshaller,	2012
394	
activemq::wireformat::openwire::marshal::v3::ActiveMQFormatMessageMarshaller,	2173
442	
activemq::wireformat::openwire::marshal::v3::ActiveMQFormatMessageMarshaller,	2241
488	
activemq::wireformat::openwire::marshal::v3::ActiveMQFormatMessageMarshaller,	2266
555	
activemq::wireformat::openwire::marshal::v3::ActiveMQFormatMessageMarshaller,	2289
584	
activemq::wireformat::openwire::marshal::v3::ActiveMQFormatMessageMarshaller,	2321
613	
activemq::wireformat::openwire::marshal::v3::ActiveMQFormatMessageMarshaller,	2348
643	
activemq::wireformat::openwire::marshal::v3::ActiveMQFormatMessageMarshaller,	2380
673	
activemq::wireformat::openwire::marshal::v3::ActiveMQFormatMessageMarshaller,	2435
703	
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller,	2659
772	
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller,	2696
880	
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller,	2731
912	
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller,	2769
1308	
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller,	
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller,	
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller,	
activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller,	
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller,	
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller,	
activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller,	
activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller,	
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller,	
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller,	
activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller,	
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller,	
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller,	
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller,	
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller,	
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller,	
activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller,	
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller,	
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller,	
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller,	
activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller,	
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller,	
activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller,	
activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller,	
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller,	

activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller, openwire::marshal::v4::ActiveMQQueue  
 2786 497  
 activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller, openwire::marshal::v4::ActiveMQStream  
 2839 563  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempNetworkBridgeFilterMarshaller, openwire::marshal::v4::ActiveMQTemp  
 2894 592  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempPartitionConsumerMarshaller, openwire::marshal::v4::ActiveMQTemp  
 3021 621  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempProducerAckMarshaller, openwire::marshal::v4::ActiveMQTemp  
 3139 647  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempProducerWithMarshaller, openwire::marshal::v4::ActiveMQTemp  
 3171 677  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempProducerWithMarshaller, openwire::marshal::v4::ActiveMQTemp  
 3209 707  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempProducerWithMarshaller, openwire::marshal::v4::BaseCommand  
 3298 780  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempProducerWithMarshaller, openwire::marshal::v4::BrokerIdMarsh  
 3324 884  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempProducerWithMarshaller, openwire::marshal::v4::BrokerInfoMar  
 3361 916  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempProducerWithMarshaller, openwire::marshal::v4::ConnectionCon  
 3405 1313  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempProducerWithMarshaller, openwire::marshal::v4::ConnectionErr  
 3498 1346  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempProducerWithMarshaller, openwire::marshal::v4::ConnectionIdM  
 3523 1378  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempProducerWithMarshaller, openwire::marshal::v4::ConnectionInfo  
 3595 1410  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempProducerWithMarshaller, openwire::marshal::v4::ConsumerCont  
 3788 1457  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempProducerWithMarshaller, openwire::marshal::v4::ConsumerIdMa  
 3949 1487  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempProducerWithMarshaller, openwire::marshal::v4::ConsumerInfoM  
 3974 1520  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempProducerWithMarshaller, openwire::marshal::v4::ControlComma  
 4126 1550  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempProducerWithMarshaller, openwire::marshal::v4::DataArrayResp  
 4166 1585  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempProducerWithMarshaller, openwire::marshal::v4::DataResponseM  
 200 1653  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempProducerWithMarshaller, openwire::marshal::v4::DestinationInfo  
 245 1795  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempProducerWithMarshaller, openwire::marshal::v4::DiscoveryEven  
 333 1829  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempProducerWithMarshaller, openwire::marshal::v4::ExceptionResp  
 375 1916  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempProducerWithMarshaller, openwire::marshal::v4::FlushCommand  
 403 2016  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempProducerWithMarshaller, openwire::marshal::v4::IntegerRespons  
 451 2177

activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	2245	3800	activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	2274	3953	activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	2297	3983	activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	2329	4118	activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	2353	4157	activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	2392	208	activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	2443	249	activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	2663	337	activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	2705	379	activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	2735	407	activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	2761	455	activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	2795	501	activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	2843	567	activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	2898	596	activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	3025	625	activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	3134	655	activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	3167	686	activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	3192	716	activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	3311	794	activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	3341	892	activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	3349	925	activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	3390	1321	activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	3486	1355	activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	3531	1386	activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueAckMapMarshaller,	3599	1418	activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller,

activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller; marshal::v5::PartialCommand  
 1465 3016  
 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller; marshal::v5::ProducerAckM  
 1495 3143  
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller; marshal::v5::ProducerIdMar  
 1528 3175  
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller; marshal::v5::ProducerInfoM  
 1558 3205  
 activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller; marshal::v5::RemoveInfoMa  
 1593 3306  
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller; marshal::v5::RemoveSubscri  
 1636 3337  
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller; marshal::v5::ReplayComm  
 1807 3370  
 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller; marshal::v5::ResponseMarsh  
 1837 3400  
 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller; marshal::v5::SessionIdMarsh  
 1912 3494  
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller; marshal::v5::SessionInfoMar  
 2025 3514  
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller; marshal::v5::ShutdownInfoM  
 2186 3590  
 activemq::wireformat::openwire::marshal::v5::JmsQueueInfoMarshaller; marshal::v5::SubscriptionInf  
 2237 3796  
 activemq::wireformat::openwire::marshal::v5::JmsQueueTopicAckMarshaller; marshal::v5::TransactionIdM  
 2258 3937  
 activemq::wireformat::openwire::marshal::v5::JmsQueueTraceMarshaller; marshal::v5::TransactionInfo  
 2305 3966  
 activemq::wireformat::openwire::marshal::v5::JmsQueueTransactionMarshaller; marshal::v5::WireFormatInfo  
 2325 4106  
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller; marshal::v5::XATransactionI  
 2357 4170  
 activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller; marshal::v6::ActiveMQBlob  
 2388 212  
 activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller; marshal::v6::ActiveMQByte  
 2439 253  
 activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller; marshal::v6::ActiveMQDest  
 2671 345  
 activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller; marshal::v6::ActiveMQMap  
 2701 387  
 activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller; marshal::v6::ActiveMQMess  
 2743 416  
 activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller; marshal::v6::ActiveMQObj  
 2765 463  
 activemq::wireformat::openwire::marshal::v5::MessageMarshaller; marshal::v6::ActiveMQQue  
 2781 510  
 activemq::wireformat::openwire::marshal::v5::MessagePurgeMarshaller; marshal::v6::ActiveMQStrea  
 2834 576  
 activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller; marshal::v6::ActiveMQTemp  
 2890 604



activemq::wireformat::openwire::marshal::v6::ActiveMQInfoQueueMarshaller	634	2313	activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller
activemq::wireformat::openwire::marshal::v6::ActiveMQInfoTopicMarshaller	664	2340	activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller
activemq::wireformat::openwire::marshal::v6::ActiveMQInfoMessageMarshaller	690	2375	activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller
activemq::wireformat::openwire::marshal::v6::ActiveMQInfoMessageMarshaller	720	2426	activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller
activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller	801	2650	activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller
activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller	896	2713	activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller
activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller	929	2722	activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller
activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller	1325	2773	activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller
activemq::wireformat::openwire::marshal::v6::ConnectionFromMarshaller	1359	2804	activemq::wireformat::openwire::marshal::v6::MessageMarshaller
activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller	1390	2851	activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller
activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller	1423	2886	activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller
activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller	1470	3007	activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller
activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller	1499	3147	activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller
activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller	1533	3179	activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller
activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller	1562	3213	activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller
activemq::wireformat::openwire::marshal::v6::DestinationResponseMarshaller	1598	3294	activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller
activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller	1641	3332	activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller
activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller	1803	3366	activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller
activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller	1817	3415	activemq::wireformat::openwire::marshal::v6::ResponseMarshaller
activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller	1899	3490	activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller
activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller	2003	3510	activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller
activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller	2164	3578	activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller
activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller	2229	3804	activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller
activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller	2270	3957	activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller
activemq::wireformat::openwire::marshal::v6::JournalTopicMarshaller	2293	3978	activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller

activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller,  
 4110  
 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller,  
 4149  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller,  
 712  
 looseMarshalBrokerError  
 activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, openwire::marshal::v1::BaseCommandMarshaller,  
 820 788  
 looseMarshalCachedObject  
 activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller,  
 821  
 activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller,  
 921  
 looseMarshalLong  
 activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, openwire::marshal::v1::ConnectionConfirmationMarshaller,  
 821 1317  
 looseMarshalNestedObject  
 activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller,  
 822  
 activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller,  
 2978  
 activemq::wireformat::openwire::OpenWireFormat, 683  
 looseMarshalObjectArray  
 activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, openwire::marshal::v1::ConsumerConfirmationMarshaller,  
 822 1462  
 looseMarshalString  
 activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller,  
 822  
 activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller,  
 1525  
 looseUnmarshal  
 activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, openwire::marshal::v1::ControlCommandMarshaller,  
 823 1554  
 activemq::wireformat::openwire::marshal::DataStreamMarshaller, openwire::marshal::v1::DataArrayResponseMarshaller,  
 1683 1590  
 activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller, openwire::marshal::v1::DataResponseMarshaller,  
 196 1658  
 activemq::wireformat::openwire::marshal::v1::ActiveMQBorrowMessageMarshaller, openwire::marshal::v1::DestinationInfoMarshaller,  
 241 1799  
 activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, openwire::marshal::v1::DiscoveryEventMarshaller,  
 329 1833  
 activemq::wireformat::openwire::marshal::v1::ActiveMQErrorMessageMarshaller, openwire::marshal::v1::ExceptionResponseMarshaller,  
 371 1921  
 activemq::wireformat::openwire::marshal::v1::ActiveMQErrorMessageMarshaller, openwire::marshal::v1::FlushCommandMarshaller,  
 399 2021  
 activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller, openwire::marshal::v1::IntegerResponseMarshaller,  
 447 2182  
 activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller, openwire::marshal::v1::JournalQueueAcknowledgeMarshaller,  
 493 2250  
 activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMessageMarshaller, openwire::marshal::v1::JournalTopicAcknowledgeMarshaller,  
 559 2279  
 activemq::wireformat::openwire::marshal::v1::ActiveMQQueueDestinationMarshaller, openwire::marshal::v1::JournalTraceMarshaller,  
 588 2302  
 activemq::wireformat::openwire::marshal::v1::ActiveMQQueueQueueMarshaller, openwire::marshal::v1::JournalTransactionMarshaller,  
 617 2333  
 activemq::wireformat::openwire::marshal::v1::ActiveMQQueueTopicMarshaller, openwire::marshal::v1::KeepAliveInfoMarshaller,

2362	4162
activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller
2397	204
activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller
2448	258
activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller
2668	341
activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller
2710	384
activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller
2740	412
activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller
2777	460
activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller
2800	506
activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller
2848	572
activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller
2903	600
activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller
3030	630
activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller
3152	660
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller
3183	695
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller
3201	725
activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller	activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller
3303	809
activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller	activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller
3320	900
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller	activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller
3353	934
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller	activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller
3411	1330
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller
3503	1338
activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller	activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller
3519	1371
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller
3587	1402
activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller	activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller
3793	1449
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller	activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller
3941	1479
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller
3970	1512
activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller	activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller
4123	1542
activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller	activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller

1577  
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller, openwire::marshal::v2::RemoveSubscriber  
 1645  
 3329  
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller, openwire::marshal::v2::ReplayCommand  
 1787  
 3358  
 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, openwire::marshal::v2::ResponseMarshaller  
 1821  
 3396  
 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller, marshal::v2::SessionIdMarshaller  
 1904  
 3483  
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller, openwire::marshal::v2::SessionInfoMarshaller  
 2008  
 3528  
 activemq::wireformat::openwire::marshal::v2::ImageResponseMarshaller, openwire::marshal::v2::ShutdownInfoMarshaller  
 2169  
 3582  
 activemq::wireformat::openwire::marshal::v2::JmsQueueInfoMarshaller, openwire::marshal::v2::SubscriptionInfoMarshaller  
 2234  
 3809  
 activemq::wireformat::openwire::marshal::v2::JmsTopicAckMarshaller, openwire::marshal::v2::TransactionIdMarshaller  
 2263  
 3945  
 activemq::wireformat::openwire::marshal::v2::JmsTopicInfoMarshaller, openwire::marshal::v2::TransactionInfoMarshaller  
 2286  
 3987  
 activemq::wireformat::openwire::marshal::v2::JmsTransactionMarshaller, marshal::v2::WireFormatInfoMarshaller  
 2317  
 4115  
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller, openwire::marshal::v2::XATransactionInfoMarshaller  
 2345  
 4153  
 activemq::wireformat::openwire::marshal::v2::LocalPrivateCommandMarshaller, marshal::v3::ActiveMQBlobMarshaller  
 2384  
 192  
 activemq::wireformat::openwire::marshal::v2::LocalTransactionMarshaller, marshal::v3::ActiveMQByteMarshaller  
 2431  
 237  
 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller, openwire::marshal::v3::ActiveMQDestinationMarshaller  
 2655  
 325  
 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller, openwire::marshal::v3::ActiveMQMapMarshaller  
 2693  
 367  
 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller, marshal::v3::ActiveMQMessageMarshaller  
 2727  
 395  
 activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller, openwire::marshal::v3::ActiveMQObjectMarshaller  
 2757  
 443  
 activemq::wireformat::openwire::marshal::v2::MessageMarshaller, openwire::marshal::v3::ActiveMQQueueMarshaller  
 2791  
 489  
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller, openwire::marshal::v3::ActiveMQStreamMarshaller  
 2831  
 555  
 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller, marshal::v3::ActiveMQTempQueueMarshaller  
 2883  
 584  
 activemq::wireformat::openwire::marshal::v2::PartialWireFormatMarshaller, openwire::marshal::v3::ActiveMQTempQueueMarshaller  
 3012  
 613  
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller, openwire::marshal::v3::ActiveMQTempQueueMarshaller  
 3131  
 643  
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller, openwire::marshal::v3::ActiveMQTextMarshaller  
 3163  
 674  
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller, openwire::marshal::v3::ActiveMQTopicMarshaller  
 3197  
 703  
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller, openwire::marshal::v3::BaseCommandMarshaller

773	2659
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller
880	2697
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller
913	2731
activemq::wireformat::openwire::marshal::v3::BrokerNameMarshaller	activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller
1309	2769
activemq::wireformat::openwire::marshal::v3::BrokerUrlMarshaller	activemq::wireformat::openwire::marshal::v3::MessageMarshaller
1342	2786
activemq::wireformat::openwire::marshal::v3::BrokerVersionMarshaller	activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller
1375	2839
activemq::wireformat::openwire::marshal::v3::BrokerWireFormatMarshaller	activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller
1406	2895
activemq::wireformat::openwire::marshal::v3::BrokerWireFormatVersionMarshaller	activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller
1453	3021
activemq::wireformat::openwire::marshal::v3::BrokerWireFormatVersion2Marshaller	activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller
1483	3139
activemq::wireformat::openwire::marshal::v3::BrokerWireFormatVersion3Marshaller	activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller
1516	3171
activemq::wireformat::openwire::marshal::v3::BrokerWireFormatVersion4Marshaller	activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller
1546	3209
activemq::wireformat::openwire::marshal::v3::BrokerWireFormatVersion5Marshaller	activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller
1581	3298
activemq::wireformat::openwire::marshal::v3::BrokerWireFormatVersion6Marshaller	activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller
1650	3324
activemq::wireformat::openwire::marshal::v3::BrokerWireFormatVersion7Marshaller	activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
1791	3362
activemq::wireformat::openwire::marshal::v3::BrokerWireFormatVersion8Marshaller	activemq::wireformat::openwire::marshal::v3::ResponseMarshaller
1825	3406
activemq::wireformat::openwire::marshal::v3::BrokerWireFormatVersion9Marshaller	activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller
1908	3499
activemq::wireformat::openwire::marshal::v3::BrokerWireFormatVersion10Marshaller	activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller
2012	3523
activemq::wireformat::openwire::marshal::v3::BrokerWireFormatVersion11Marshaller	activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller
2173	3595
activemq::wireformat::openwire::marshal::v3::BrokerWireFormatVersion12Marshaller	activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller
2242	3789
activemq::wireformat::openwire::marshal::v3::BrokerWireFormatVersion13Marshaller	activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller
2267	3949
activemq::wireformat::openwire::marshal::v3::BrokerWireFormatVersion14Marshaller	activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller
2290	3975
activemq::wireformat::openwire::marshal::v3::BrokerWireFormatVersion15Marshaller	activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller
2321	4127
activemq::wireformat::openwire::marshal::v3::BrokerWireFormatVersion16Marshaller	activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller
2349	4166
activemq::wireformat::openwire::marshal::v3::BrokerWireFormatVersion17Marshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller
2380	200
activemq::wireformat::openwire::marshal::v3::BrokerWireFormatVersion18Marshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller
2435	245
activemq::wireformat::openwire::marshal::v3::BrokerWireFormatVersion19Marshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller

333	1829
activemq::wireformat::openwire::marshal::v4::ActiveMQInfoMessageMarshaller	1917
375	1917
activemq::wireformat::openwire::marshal::v4::ActiveMQInfoMessageMarshaller	2017
403	2017
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	2178
451	2178
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	2246
497	2246
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	2275
564	2275
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	2298
592	2298
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	2329
622	2329
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	2353
647	2353
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	2393
678	2393
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	2444
708	2444
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	2663
781	2663
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	2705
884	2705
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	2735
917	2735
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	2761
1313	2761
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	2795
1347	2795
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	2843
1379	2843
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	2899
1410	2899
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	3026
1457	3026
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	3135
1487	3135
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	3167
1520	3167
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	3192
1550	3192
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	3311
1585	3311
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	3341
1654	3341
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	3349
1795	3349
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	

3391	1322
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller,openwire::marshal::v5::ConnectionErrorMarshaller,	
3487	1355
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller,openwire::marshal::v5::ConnectionIdMarshaller,	
3532	1387
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller,openwire::marshal::v5::ConnectionInfoMarshaller,	
3599	1419
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller,openwire::marshal::v5::ConsumerControlMarshaller,	
3801	1466
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller,openwire::marshal::v5::ConsumerIdMarshaller,	
3953	1495
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller,openwire::marshal::v5::ConsumerInfoMarshaller,	
3983	1529
activemq::wireformat::openwire::marshal::v4::WireFormatMarshaller,openwire::marshal::v5::ControlCommandMarshaller,	
4119	1559
activemq::wireformat::openwire::marshal::v4::ByteArrayResponseMarshaller,openwire::marshal::v5::DataArrayResponseMarshaller,	
4158	1594
activemq::wireformat::openwire::marshal::v5::ActiveMQDeflatingMessageMarshaller,openwire::marshal::v5::DataResponseMarshaller,	
209	1637
activemq::wireformat::openwire::marshal::v5::ActiveMQDeflatingMessageMarshaller,openwire::marshal::v5::DestinationInfoMarshaller,	
250	1808
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller,openwire::marshal::v5::DiscoveryEventMarshaller,	
337	1837
activemq::wireformat::openwire::marshal::v5::ActiveMQDeflatingMessageMarshaller,openwire::marshal::v5::ExceptionResponseMarshaller,	
379	1913
activemq::wireformat::openwire::marshal::v5::ActiveMQDeflatingMessageMarshaller,openwire::marshal::v5::FlushCommandMarshaller,	
408	2025
activemq::wireformat::openwire::marshal::v5::ActiveMQDeflatingMessageMarshaller,openwire::marshal::v5::IntegerResponseMarshaller,	
455	2186
activemq::wireformat::openwire::marshal::v5::ActiveMQDeflatingMessageMarshaller,openwire::marshal::v5::JournalQueueAckMarshaller,	
502	2238
activemq::wireformat::openwire::marshal::v5::ActiveMQDeflatingMessageMarshaller,openwire::marshal::v5::JournalTopicAckMarshaller,	
568	2259
activemq::wireformat::openwire::marshal::v5::ActiveMQDeflatingMessageMarshaller,openwire::marshal::v5::JournalTraceMarshaller,	
596	2306
activemq::wireformat::openwire::marshal::v5::ActiveMQDeflatingMessageMarshaller,openwire::marshal::v5::JournalTransactionMarshaller,	
626	2325
activemq::wireformat::openwire::marshal::v5::ActiveMQDeflatingMessageMarshaller,openwire::marshal::v5::KeepAliveInfoMarshaller,	
656	2357
activemq::wireformat::openwire::marshal::v5::ActiveMQDeflatingMessageMarshaller,openwire::marshal::v5::LastPartialCommandMarshaller,	
686	2389
activemq::wireformat::openwire::marshal::v5::ActiveMQDeflatingMessageMarshaller,openwire::marshal::v5::LocalTransactionIdMarshaller,	
716	2440
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller,openwire::marshal::v5::MessageAckMarshaller,	
795	2672
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller,openwire::marshal::v5::MessageDispatchMarshaller,	
892	2701
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller,openwire::marshal::v5::MessageDispatchNotificationMarshaller,	
925	2744
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller,openwire::marshal::v5::MessageIdMarshaller,	

2765  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v6::ActiveMQQueueMessageMarshaller 510  
 2782  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v6::ActiveMQQueueMessageMarshaller 576  
 2835  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v6::ActiveMQQueueMessageMarshaller 604  
 2891  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v6::ActiveMQQueueMessageMarshaller 634  
 3017  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v6::ActiveMQQueueMessageMarshaller 664  
 3143  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v6::ActiveMQQueueMessageMarshaller 691  
 3175  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v6::ActiveMQQueueMessageMarshaller 720  
 3205  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v6::ActiveMQQueueMessageMarshaller 802  
 3307  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v6::ActiveMQQueueMessageMarshaller 896  
 3337  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v6::ActiveMQQueueMessageMarshaller 930  
 3370  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v6::ActiveMQQueueMessageMarshaller 1326  
 3401  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v6::ActiveMQQueueMessageMarshaller 1359  
 3495  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v6::ActiveMQQueueMessageMarshaller 1391  
 3515  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v6::ActiveMQQueueMessageMarshaller 1423  
 3591  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v6::ActiveMQQueueMessageMarshaller 1470  
 3797  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v6::ActiveMQQueueMessageMarshaller 1499  
 3937  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v6::ActiveMQQueueMessageMarshaller 1533  
 3966  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v6::ActiveMQQueueMessageMarshaller 1563  
 4107  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v6::ActiveMQQueueMessageMarshaller 1598  
 4170  
 activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMessageMarshaller, marshal::v6::ActiveMQQueueMessageMarshaller 1641  
 213  
 activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMessageMarshaller, marshal::v6::ActiveMQQueueMessageMarshaller 1804  
 254  
 activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMessageMarshaller, marshal::v6::ActiveMQQueueMessageMarshaller 1817  
 345  
 activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMessageMarshaller, marshal::v6::ActiveMQQueueMessageMarshaller 1900  
 388  
 activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMessageMarshaller, marshal::v6::ActiveMQQueueMessageMarshaller 2004  
 416  
 activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMessageMarshaller, marshal::v6::ActiveMQQueueMessageMarshaller



2165  
 activemq::wireformat::openwire::marshal::v6::QueueAckMarshaller, 3578  
 2230  
 activemq::wireformat::openwire::marshal::v6::QueueInfoMarshaller, 3805  
 2271  
 activemq::wireformat::openwire::marshal::v6::TopicAckMarshaller, 3957  
 2294  
 activemq::wireformat::openwire::marshal::v6::TopicInfoMarshaller, 3979  
 2313  
 activemq::wireformat::openwire::marshal::v6::TransactionMarshaller, 4111  
 2340  
 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller, 4149  
 2376  
 activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller, 4251  
 2427  
 looseUnmarshalByteArray, 4251  
 activemq::wireformat::openwire::marshal::v6::LogIdMarshaller, 4251  
 2651  
 activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller, 824  
 2714  
 looseUnmarshalCachedObject, 824  
 2723  
 activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller, 824  
 2773  
 looseUnmarshalConstByteArray, 824  
 2804  
 activemq::wireformat::openwire::marshal::v6::MessageMarshaller, 824  
 2852  
 looseUnmarshalNestedObject, 825  
 2887  
 activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller, 825  
 3008  
 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, 2979  
 3148  
 looseUnmarshalString, 2979  
 3179  
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller, 2979  
 3214  
 looseUnmarshalString, 2979  
 3294  
 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller, 2979  
 3333  
 decaf::lang::Integer, 2150  
 3366  
 decaf::lang::Long, 2502  
 3416  
 activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller, 2537  
 3491  
 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller, 2538  
 3511  
 activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller, 2538  
 3578  
 activemq::wireformat::openwire::marshal::v6::ResponseCommandMarshaller, 2538  
 3805  
 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller, 2540  
 3957  
 activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 2540  
 3979  
 activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller, 3104  
 4111  
 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller, 3104  
 4149  
 activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller, 3104  
 4251  
 activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller, 3104

activemq::util::PrimitiveValueNode::PrimitiveValueNode, 3098  
 mark  
   decaf::io::BufferedInputStream, 946  
   decaf::io::ByteArrayInputStream, 1043  
   decaf::io::FilterInputStream, 1955  
   decaf::io::InputStream, 2109  
   decaf::io::PushbackInputStream, 3235  
   decaf::io::Reader, 3256  
   decaf::nio::Buffer, 941  
   decaf::util::logging::SimpleLogger, 3607  
   decaf::util::zip::InflaterInputStream, 2102  
 Markblock  
   decaf::util::logging, 155  
 MarkBlockLogger  
   decaf::util::logging::MarkBlockLogger, 2563  
 markSupported  
   decaf::io::BufferedInputStream, 947  
   decaf::io::ByteArrayInputStream, 1044  
   decaf::io::FilterInputStream, 1955  
   decaf::io::InputStream, 2109  
   decaf::io::PushbackInputStream, 3235  
   decaf::io::Reader, 3257  
   decaf::util::zip::InflaterInputStream, 2103  
 marshal  
   activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3092  
   activemq::wireformat::openwire::OpenWireFormat, 2979  
   activemq::wireformat::openwire::utils::BooleanStream, 864, 865  
   activemq::wireformat::stomp::StompWireFormat, 3753  
   activemq::wireformat::WireFormat, 4090  
 marshalledProperties  
   activemq::commands::Message, 2613  
 marshalledSize  
   activemq::wireformat::openwire::utils::BooleanStream, 865  
 MarshallingSupport  
   activemq::util::MarshallingSupport, 2572  
 marshalList  
   activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3092  
 marshalMap  
   activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3093  
 marshalPrimitive  
   activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3093  
   marshalPrimitiveList  
   activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3093  
   marshalPrimitiveMap  
   activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3094  
   masterBroker  
   activemq::commands::BrokerInfo, 910  
 MATCH  
   inflate.h, 4627  
   match\_available  
   internal\_state, 2191  
   match\_length  
   internal\_state, 2191  
   match\_start  
   internal\_state, 2191  
   matches  
   internal\_state, 2191  
 Math  
   decaf::lang::Math, 2578  
 max  
   decaf::lang::Math, 2581, 2582  
 MAX\_BITS  
   deflate.h, 4623  
   max\_chain\_length  
   max\_code  
   tree\_desc\_s, 4018  
 MAX\_DIST  
   deflate.h, 4623  
   max\_insert\_length  
   deflate.h, 4623  
   max\_lazy\_match  
   internal\_state, 2191  
 MAX\_MATCH  
   zutil.h, 4639  
 MAX\_MEM\_LEVEL  
   zconf.h, 4632  
 MAX\_PREFETCH\_SIZE  
   activemq::core::policies::DefaultPrefetchPolicy, 1730  
 MAX\_PRIORITY  
   activemq::lang::Marshaller, 4886  
 MAX\_RADIX  
   decaf::lang::Character, 1134  
 MAX\_VALUE  
   decaf::lang::Byte, 979  
   decaf::lang::Character, 1134

- decaf::lang::Double, 1853
- decaf::lang::Float, 1973
- decaf::lang::Integer, 2159
- decaf::lang::Long, 2510
- decaf::lang::Short, 3548
- MAX\_WBITS
- zconf.h, 4632
- maximumPendingMessageLimit
- activemq::commands::ConsumerInfo, 1509
- MEM
- inflate.h, 4627
- MemoryUsage
- activemq::util::MemoryUsage, 2593
- MESSAGE
- activemq::wireformat::stomp::StompCommandConstants, 3740
- Message
- activemq::commands::Message, 2601
- message
- activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3267
- activemq::commands::JournalTrace, 2283
- activemq::commands::MessageDispatch, 2683
- decaf::lang::Exception, 1893
- MessageAck
- activemq::commands::MessageAck, 2644
- messageAck
- activemq::commands::JournalQueueAck, 2227
- MessageAckMarshaller
- activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller, 2667
- activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller, 2654
- activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller, 2658
- activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller, 2662
- activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller, 2671
- activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller, 2650
- messageCount
- activemq::commands::MessageAck, 2644
- MessageDispatch
- activemq::commands::MessageDispatch, 2680
- MessageDispatchChannel
- activemq::core::MessageDispatchChannel, 2685
- MessageDispatchMarshaller
- activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2709
- activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller, 2692
- activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller, 2696
- activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller, 2704
- activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller, 2700
- activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller, 2713
- MessageDispatchNotification
- activemq::commands::MessageDispatchNotification, 2717
- MessageDispatchNotificationMarshaller
- activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller, 2739
- activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller, 2726
- activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller, 2730
- activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller, 2734
- activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller, 2743
- activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller, 2722
- MessageEOFException
- activemq::wireformat::openwire::marshal::v1::MessageEOFException, 2748
- MessageFormatException
- activemq::wireformat::openwire::marshal::v2::MessageFormatException, 2750
- MessageId
- activemq::commands::MessageId, 2752
- messageId
- activemq::commands::JournalTopicAck, 2256
- activemq::commands::Message, 2613
- activemq::commands::MessageDispatchNotification, 2720
- activemq::commands::MessagePull, 2828
- MessageIdMarshaller
- activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 2776
- activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller, 2756

activemq::wireformat::openwire::marshaller::v3::MessageIdMarshaller, 2768  
 activemq::wireformat::openwire::marshaller::v4::MessageIdMarshaller, 2760  
 activemq::wireformat::openwire::marshaller::v5::MessageIdMarshaller, 2764  
 activemq::wireformat::openwire::marshaller::v6::MessageIdMarshaller, 2772  
 MessageMarshaller  
 activemq::wireformat::openwire::marshaller::v1::MessageMarshaller, 2799  
 activemq::wireformat::openwire::marshaller::v2::MessageMarshaller, 2790  
 activemq::wireformat::openwire::marshaller::v3::MessageMarshaller, 2786  
 activemq::wireformat::openwire::marshaller::v4::MessageMarshaller, 2795  
 activemq::wireformat::openwire::marshaller::v5::MessageMarshaller, 2781  
 activemq::wireformat::openwire::marshaller::v6::MessageMarshaller, 2804  
 MessageNotReadableException  
 cms::MessageNotReadableException, 2808  
 MessageNotWriteableException  
 cms::MessageNotWriteableException, 2809  
 MessagePropertyInterceptor  
 activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2819  
 MessagePull  
 activemq::commands::MessagePull, 2825  
 MessagePullMarshaller  
 activemq::wireformat::openwire::marshaller::v1::MessagePullMarshaller, 2847  
 activemq::wireformat::openwire::marshaller::v2::MessagePullMarshaller, 2830  
 activemq::wireformat::openwire::marshaller::v3::MessagePullMarshaller, 2838  
 activemq::wireformat::openwire::marshaller::v4::MessagePullMarshaller, 2842  
 activemq::wireformat::openwire::marshaller::v5::MessagePullMarshaller, 2834  
 activemq::wireformat::openwire::marshaller::v6::MessagePullMarshaller, 2851  
 messageSequenceId  
 activemq::commands::JournalTopicAck, 2256  
 method  
 internal\_state, 2191  
 MethodMessageIdMarshaller, 2768  
 activemq::commands::BrokerError::StackTraceElement, 3686  
 MICROSECONDS  
 decaf::util::concurrent::TimeUnit, 3929  
 MILLISECONDS  
 decaf::util::concurrent::TimeUnit, 3929  
 min  
 decaf::lang::Math, 2582–2584  
 MIN\_LENGTH  
 deflate.h, 4623  
 MIN\_MAX  
 zutil.h, 4639  
 MIN\_PRIORITY  
 decaf::lang::Thread, 3886  
 MIN\_RADIX  
 decaf::lang::Character, 1134  
 MIN\_VALUE  
 decaf::lang::Byte, 979  
 decaf::lang::Character, 1134  
 decaf::lang::Double, 1853  
 decaf::lang::Float, 1973  
 decaf::lang::Integer, 2159  
 decaf::lang::Long, 2510  
 decaf::lang::Short, 3548  
 MINUTES  
 decaf::util::concurrent::TimeUnit, 3929  
 MockTransport  
 MessagePropertyInterceptor, 2819  
 activemq::transport::mock::MockTransport, 2857  
 mode  
 gz\_state, 2041  
 inflate\_state, 2087  
 modifiedUtf8ToAscii  
 activemq::util::MarshallingSupport, 2573  
 msg  
 gz\_state, 2041  
 MessagePullMarshaller, 2838  
 z\_stream\_s, 4175  
 Mutex  
 MessagePullMarshaller, 2842  
 decaf::util::concurrent::Mutex, 2868  
 mutex  
 MessagePullMarshaller, 2834  
 decaf::util::AbstractCollection, 173  
 MessagePullMarshaller, 2851  
 ConditionHandle, 1291  
 decaf::util::concurrent::MutexHandle, 2872  
 MutexHandle  
 decaf::util::concurrent::MutexHandle, 2872

- NAME
  - inflate.h, 4627
- name
  - gz\_header\_s, 2039
- name\_max
  - gz\_header\_s, 2039
- NAME\_STATE
  - deflate.h, 4623
- nameUUIDFromBytes
  - decaf::util::UUID, 4084, 4085
- NaN
  - decaf::lang::Double, 1853
  - decaf::lang::Float, 1973
- NANOSECONDS
  - decaf::util::concurrent::TimeUnit, 3929
- nanoTime
  - decaf::lang::System, 3845
- narrow
  - activemq::transport::failover::FailoverTransport, 1937
  - activemq::transport::IOTransport, 2217
  - activemq::transport::mock::MockTransport, 2860
  - activemq::transport::Transport, 3999
  - activemq::transport::TransportFilter, 4009
- ncode
  - inflate\_state, 2087
- ndist
  - inflate\_state, 2087
- needsDictionary
  - decaf::util::zip::Inflater, 2093
- needsInput
  - decaf::util::zip::Deflater, 1764
  - decaf::util::zip::Inflater, 2093
- NEGATIVE\_INFINITY
  - decaf::lang::Double, 1853
  - decaf::lang::Float, 1973
- Network
  - decaf::internal::net::Network, 2876
- NetworkBridgeFilter
  - activemq::commands::NetworkBridgeFilter, 2878
- NetworkBridgeFilterMarshaller
  - activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller, 2902
  - activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller, 2882
  - activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller, 2894
- activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller, 2898
- activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller, 2890
- activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller, 2886
- networkBrokerId
  - activemq::commands::NetworkBridgeFilter, 2880
- networkConnection
  - activemq::commands::BrokerInfo, 910
- networkConsumerPath
  - activemq::commands::ConsumerInfo, 1509
- networkProperties
  - activemq::commands::BrokerInfo, 910
- networkSubscription
  - activemq::commands::ConsumerInfo, 1509
- networkTTL
  - activemq::commands::NetworkBridgeFilter, 2880
- NEW
  - decaf::lang::Thread, 3880
- newCondition
  - decaf::util::concurrent::locks::Lock, 2455
  - decaf::util::concurrent::locks::ReentrantLock, 3277
- newThread
  - decaf::util::concurrent::ThreadFactory, 3887
- next
  - activemq::transport::TransportFilter, 4013
  - decaf::security::SecureRandom, 3427
  - decaf::util::Iterator, 2223
  - decaf::util::Random, 3247
  - gz\_state, 2041
  - inflate\_state, 2087
- next\_in
  - z\_stream\_s, 4175
- next\_out
  - z\_stream\_s, 4175
- nextBoolean
  - decaf::util::Random, 3248
- nextByte
  - decaf::util::Random, 3248
- nextDouble
  - decaf::util::Random, 3248, 3249
- nextInt
  - decaf::util::Random, 3249

- nextFloat
  - decaf::util::Random, 3249
- nextGaussian
  - decaf::util::Random, 3249
- nextIndex
  - decaf::util::ListIterator, 2418
- nextInt
  - decaf::util::Random, 3250
- nextLong
  - decaf::util::Random, 3250
- nextMessage
  - activemq::core::ActiveMQQueueBrowser, 486
  - cms::MessageEnumeration, 2747
- nextToken
  - decaf::util::StringTokenizer, 3780, 3781
- nice\_match
  - internal\_state, 2191
- nlen
  - inflate\_state, 2087
- NO\_COMPRESSION
  - decaf::util::zip::Deflater, 1769
- NO\_MAXIMUM\_REDELIVERIES
  - activemq::core::RedeliveryPolicy, 3272
- node
  - decaf::util::UUID, 4085
- noLocal
  - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3267
  - activemq::commands::ConsumerInfo, 1509
- NON\_PERSISTENT
  - cms::DeliveryMode, 1776
- noRangeAcks
  - activemq::commands::ConsumerInfo, 1509
- NORM\_PRIORITY
  - decaf::lang::Thread, 3886
- normalize
  - decaf::net::URI, 4040
- NoRouteToHostException
  - decaf::net::NoRouteToHostException, 2905, 2906
- NoSuchAlgorithmException
  - decaf::security::NoSuchAlgorithmException, 2908, 2909
- NoSuchElementException
  - decaf::lang::exceptions::NoSuchElementException, 2911, 2912
- NoSuchProviderException
  - decaf::security::NoSuchProviderException, 2913, 2914
- notify
  - activemq::core::MessageDispatchChannel, 2687
  - decaf::internal::util::concurrent::ConditionImpl, 1292
  - decaf::internal::util::concurrent::SynchronizableImpl, 3823
  - decaf::io::InputStream, 2110
  - decaf::io::OutputStream, 2995
  - decaf::util::AbstractCollection, 167
  - decaf::util::concurrent::ConcurrentStlMap, 1274
  - decaf::util::concurrent::Mutex, 2868
  - decaf::util::concurrent::Synchronizable, 3813
  - decaf::util::StlMap, 3717
  - decaf::util::StlQueue, 3727
- notifyAll
  - activemq::core::MessageDispatchChannel, 2687
  - decaf::internal::util::concurrent::ConditionImpl, 1292
  - decaf::internal::util::concurrent::SynchronizableImpl, 3823
  - decaf::io::InputStream, 2110
  - decaf::io::OutputStream, 2995
  - decaf::util::AbstractCollection, 167
  - decaf::util::concurrent::ConcurrentStlMap, 1274
  - decaf::util::concurrent::Mutex, 2868
  - decaf::util::concurrent::Synchronizable, 3814
  - decaf::util::StlMap, 3717
  - decaf::util::StlQueue, 3727
- Null
  - decaf::util::logging, 155
- NULL\_TYPE
  - activemq::util::PrimitiveValueNode, 3104
  - activemq::wireformat::openwire::OpenWireFormat, 2984
- NullPointerException
  - decaf::lang::exceptions::NullPointerException, 2916, 2917
- NUM\_OPTIONS
  - activemq::core::ActiveMQConstants, 299
- NUM\_PARAMS

- activemq::core::ActiveMQConstants, 300
- NumberFormatException
  - decaf::lang::exceptions::NumberFormatException, 2067
  - 2922, 2923
- numberOfLeadingZeros
  - decaf::lang::Integer, 2150
  - decaf::lang::Long, 2502
- numberOfTrailingZeros
  - decaf::lang::Integer, 2151
  - decaf::lang::Long, 2503
- numWaiting
  - decaf::util::concurrent::ConditionHandle, 1291
- numWake
  - decaf::util::concurrent::ConditionHandle, 1291
- objectId
  - activemq::commands::RemoveInfo, 3287
- OF
  - deflate.h, 4623
  - gzguts.h, 4625
  - inffast.h, 4626
  - inftrees.h, 4628
  - zconf.h, 4632
  - zlib.h, 4636
  - zutil.h, 4639
- OFF
  - decaf::util::logging::Level, 2408
- Off
  - decaf::util::logging, 155
- offer
  - decaf::util::concurrent::BlockingQueue, 853
  - decaf::util::concurrent::SynchronousQueue, 3833, 3834
  - decaf::util::PriorityQueue, 3121
  - decaf::util::Queue, 3241
- offset
  - decaf::internal::nio::CharArrayBuffer, 1148
  - inflate\_state, 2087
- onCommand
  - activemq::core::ActiveMQConnection, 274
  - activemq::transport::correlator::ResponseCorrelator, 3386
  - activemq::transport::DefaultTransportListener, 1758
- onException
  - activemq::core::ActiveMQConnection, 275
  - activemq::transport::DefaultTransportListener, 1758
  - activemq::transport::failover::BackupTransport, 760
  - activemq::transport::failover::FailoverTransportListener, 1946
  - activemq::transport::inactivity::InactivityMonitor, 2068
  - activemq::transport::TransportFilter, 4010
  - activemq::transport::TransportListener, 4014
  - activemq::transport::correlator::ResponseCorrelatorExceptionListener, 1894
- onMessage
  - activemq::transport::DefaultTransportListener, 1758
  - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2987
- onProducerAck
  - activemq::transport::failover::FailoverTransportListener, 1946
  - activemq::transport::inactivity::InactivityMonitor, 2067
  - activemq::transport::logging::LoggingTransport, 2478
  - activemq::transport::mock::InternalCommandListener, 2193
  - activemq::transport::TransportFilter, 4009
  - activemq::transport::TransportListener, 4014
  - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2987
- oneway
  - activemq::core::ActiveMQConnection, 274
  - activemq::core::ActiveMQSession, 528
  - activemq::transport::correlator::ResponseCorrelator, 3386
  - activemq::transport::failover::FailoverTransport, 1937
  - activemq::transport::inactivity::InactivityMonitor, 2068
  - activemq::transport::IOTransport, 2217
  - activemq::transport::logging::LoggingTransport, 2478
  - activemq::transport::mock::MockTransport, 2860
  - activemq::transport::Transport, 3999
  - activemq::transport::TransportFilter, 4010
  - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2987

activemq::core::ActiveMQProducer, 470	decaf::internal::net::ssl::openssl::OpenSSLServerSocket,
onPropertiesReset	2932
decaf::util::logging::PropertiesChangeListener, 3227	OpenSSLServerSocketFactory
onPropertyChanged	2938
decaf::util::logging::PropertiesChangeListener, 3227	decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory
onResponse	2927
activemq::state::Tracked, 3932	decaf::internal::net::ssl::openssl::OpenSSLSocket,
onSend	2946
activemq::commands::ActiveMQBytes, 221	OpenSSLSocketException
activemq::commands::ActiveMQMessageTemplate, 431	decaf::internal::net::ssl::openssl::OpenSSLSocketException,
activemq::commands::ActiveMQStreamMessage, 540	OpenSSLSocketFactory
activemq::commands::Message, 2608	decaf::internal::net::ssl::openssl::OpenSSLContextSpi,
onTaskComplete	2927
decaf::util::concurrent::TaskListener, 3848	decaf::internal::net::ssl::openssl::OpenSSLSocketFactory,
onTaskCompleted	2962
decaf::util::concurrent::PooledThreadListener, 3059	OpenSSLSocketInputStream
decaf::util::concurrent::ThreadPool, 3892	decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream
onTaskException	2967
decaf::util::concurrent::PooledThreadListener, 3059	OpenSSLSocketOutputStream
decaf::util::concurrent::TaskListener, 3848	decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream
decaf::util::concurrent::ThreadPool, 3893	2970
onTaskStarted	OpenWireFormat
decaf::util::concurrent::PooledThreadListener, 3059	activemq::wireformat::openwire::OpenWireFormat,
decaf::util::concurrent::TaskListener, 3848	2974
decaf::util::concurrent::ThreadPool, 3893	OpenWireFormatFactory
onTaskStarted	activemq::wireformat::openwire::OpenWireFormatFactory,
decaf::util::concurrent::PooledThreadListener, 3059	2985
decaf::util::concurrent::ThreadPool, 3893	OpenWireFormatNegotiator
onTransportException	activemq::wireformat::openwire::OpenWireFormatNegotiator,
activemq::transport::correlator::ResponseCorrelator, 3387	2986
activemq::wireformat::openwire::OpenWireFormatNegotiator, 2988	OpenWireResponseBuilder
op	activemq::wireformat::openwire::OpenWireResponseBuilder,
code, 1216	2990
opaque	operationType
z_stream_s, 4175	activemq::commands::DestinationInfo,
OPEN_FAILURE	1784
decaf::util::logging::ErrorManager, 1886	operator<
OpenSSLContextSpi	activemq::commands::BrokerId, 877
decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2926	activemq::commands::ConnectionId,
OpenSSLServerSocket	1368
	activemq::commands::ConsumerId, 1476
	activemq::commands::LocalTransactionId,
	2424
	activemq::commands::MessageId, 2754
	activemq::commands::ProducerId, 3160
	activemq::commands::SessionId, 3480



- activemq::commands::TransactionId, 3935
- activemq::commands::XATransactionId, 4146
- decaf::lang::Boolean, 858, 859
- decaf::lang::Byte, 974, 975
- decaf::lang::Character, 1132
- decaf::lang::Comparable, 1250
- decaf::lang::Double, 1849
- decaf::lang::Float, 1969
- decaf::lang::Integer, 2151, 2152
- decaf::lang::Long, 2503
- decaf::lang::Short, 3544
- decaf::net::URI, 4040
- decaf::nio::ByteBuffer, 1068
- decaf::nio::CharBuffer, 1159
- decaf::nio::DoubleBuffer, 1873
- decaf::nio::FloatBuffer, 1993
- decaf::nio::IntBuffer, 2138
- decaf::nio::LongBuffer, 2530
- decaf::nio::ShortBuffer, 3568
- decaf::util::concurrent::TimeUnit, 3924
- decaf::util::Date, 1722
- decaf::util::logging::Level, 2406
- decaf::util::UUID, 4085
- operator\*
  - decaf::lang::Pointer, 3037
- operator()
  - decaf::lang::ArrayPointerComparator, 745
  - decaf::lang::PointerComparator, 3041
  - decaf::util::Comparator, 1252
  - decaf::util::comparators::Less, 2401
  - std::less< decaf::lang::ArrayPointer< T > >, 2402
  - std::less< decaf::lang::Pointer< T > >, 2403
- operator->
  - decaf::lang::Pointer, 3038
- operator=
  - activemq::cmsutil::CachedConsumer, 1100
  - activemq::cmsutil::CachedProducer, 1105
  - activemq::cmsutil::CmsAccessor, 1187
  - activemq::cmsutil::CmsDestinationAccessor, 1190
  - activemq::cmsutil::CmsTemplate, 1208
  - activemq::cmsutil::CmsTemplate::ProducerExecutor, 3156
- activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3267
- activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 3373
- activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 3374
- activemq::cmsutil::CmsTemplate::SendExecutor, 3446
- activemq::cmsutil::DynamicDestinationResolver, 1879
- activemq::cmsutil::PooledSession, 3054
- activemq::cmsutil::ResourceLifecycleManager, 3379
- activemq::cmsutil::SessionPool, 3535
- activemq::commands::BrokerId, 877
- activemq::commands::ConnectionId, 1368
- activemq::commands::ConsumerId, 1476
- activemq::commands::LocalTransactionId, 2424
- activemq::commands::MessageId, 2754
- activemq::commands::ProducerId, 3160
- activemq::commands::SessionId, 3480
- activemq::commands::TransactionId, 3935
- activemq::commands::XATransactionId, 4146
- activemq::library::ActiveMQCPP, 312
- activemq::util::PrimitiveValueNode, 3111
- decaf::lang::ArrayPointer, 742
- decaf::lang::Exception, 1892
- decaf::lang::Pointer, 3038
- decaf::util::AbstractCollection, 167
- decaf::util::Date, 1722
- decaf::util::logging::LogManager, 2485
- decaf::util::PriorityQueue, 3122
- decaf::util::Properties, 3223
- operator==
  - activemq::commands::BrokerId, 877
  - activemq::commands::ConnectionId, 1368
  - activemq::commands::ConsumerId, 1476
  - activemq::commands::LocalTransactionId, 2424
  - activemq::commands::MessageId, 2754
  - activemq::commands::ProducerId, 3160
  - activemq::commands::SessionId, 3480
  - activemq::commands::TransactionId, 3935

- activemq::commands::XATransactionId, 4146
- activemq::util::PrimitiveValueNode, 3111
- decaf::lang, 144
- decaf::lang::ArrayPointer, 742, 744
- decaf::lang::Boolean, 859
- decaf::lang::Byte, 975
- decaf::lang::Character, 1133
- decaf::lang::Comparable, 1250
- decaf::lang::Double, 1849, 1850
- decaf::lang::Float, 1970
- decaf::lang::Integer, 2152
- decaf::lang::Long, 2504
- decaf::lang::Pointer, 3039, 3040
- decaf::lang::Short, 3544, 3545
- decaf::net::URI, 4041
- decaf::nio::ByteBuffer, 1068
- decaf::nio::CharBuffer, 1160
- decaf::nio::DoubleBuffer, 1873
- decaf::nio::FloatBuffer, 1993
- decaf::nio::IntBuffer, 2138
- decaf::nio::LongBuffer, 2530
- decaf::nio::ShortBuffer, 3568
- decaf::util::concurrent::TimeUnit, 3924
- decaf::util::Date, 1722
- decaf::util::logging::Level, 2406
- decaf::util::UUID, 4086
- operator[]
  - activemq::wireformat::openwire::utils::HexTable, 300
  - 2048, 2049
  - decaf::internal::util::ByteArrayAdapter, 996
  - decaf::lang::ArrayPointer, 742
- opt\_len
  - internal\_state, 2191
- optimizedAcknowledge
  - activemq::commands::ConsumerInfo, 1509
- options
  - activemq::commands::ActiveMQDestination, 323
- ordered
  - activemq::commands::ActiveMQDestination, 323
- orderedTarget
  - activemq::commands::ActiveMQDestination, 323
- originalDestination
  - activemq::commands::Message, 2613
- originalTransactionId
- activemq::commands::Message, 2613
- OS
- os
  - gz\_header\_s, 2039
- OS\_CODE
  - zutil.h, 4639
- out
  - decaf::io::FileDescriptor, 1949
  - gz\_state, 2041
- OutputStream
  - decaf::io::OutputStream, 2993
- outputStream
  - decaf::io::FilterOutputStream, 1961
- OutputStreamWriter
  - decaf::io::OutputStreamWriter, 3000
- own
  - decaf::io::FilterInputStream, 1957
  - decaf::io::FilterOutputStream, 1961
- ownDeflater
  - decaf::util::zip::DeflaterOutputStream, 1774
- ownInflater
  - decaf::util::zip::InflaterInputStream, 2105
- PARAM\_CLIENTID
  - activemq::core::ActiveMQConstants, 300
- PARAM\_PASSWORD
  - activemq::core::ActiveMQConstants, 300
- PARAM\_USERNAME
  - activemq::core::ActiveMQConstants, 300
- parent
  - activemq::cmsutil::CmsTemplate::ProducerExecutor, 3156
  - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3267
- park
  - decaf::util::concurrent::locks::LockSupport, 2459
- parkNanos
  - decaf::util::concurrent::locks::LockSupport, 2460
- parkUntil
  - decaf::util::concurrent::locks::LockSupport, 2460
- parse

- decaf::internal::util::HexStringParser, 2047
- decaf::util::logging::Level, 2406
- parseAuthority
  - decaf::internal::net::URIHelper, 4051
- parseBoolean
  - decaf::lang::Boolean, 860
- parseByte
  - decaf::lang::Byte, 976
- parseComposite
  - activemq::util::URISupport, 4058
- parseDouble
  - decaf::internal::util::HexStringParser, 2047
  - decaf::lang::Double, 1850
- parseFloat
  - decaf::internal::util::HexStringParser, 2047
  - decaf::lang::Float, 1970
- parseInt
  - decaf::lang::Integer, 2153
- parseLong
  - decaf::lang::Long, 2504, 2505
- parseQuery
  - activemq::util::URISupport, 4058, 4059
- parseServerAuthority
  - decaf::net::URI, 4041
- parseShort
  - decaf::lang::Short, 3545, 3546
- parseURI
  - decaf::internal::net::URIHelper, 4051
- parseURL
  - activemq::util::URISupport, 4059
- PartialCommand
  - activemq::commands::PartialCommand, 3003
- PartialCommandMarshaller
  - activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller, 3029
  - activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller, 3011
  - activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller, 3020
  - activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller, 3025
  - activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, 3016
  - activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, 3007
- password
- activemq::commands::ConnectionInfo, 1399
- path
  - gz\_state, 2041
- peek
  - activemq::core::MessageDispatchChannel, 2687
  - decaf::internal::util::TimerTaskHeap, 3919
  - decaf::util::concurrent::SynchronousQueue, 3834
  - decaf::util::PriorityQueue, 3122
  - decaf::util::Queue, 3242
- peerBrokerInfos
  - activemq::commands::BrokerInfo, 910
- pending
  - internal\_state, 2191
- pending\_buf
  - internal\_state, 2191
- pending\_buf\_size
  - internal\_state, 2191
- pending\_out
  - internal\_state, 2191
- PERSISTENT
  - cms::DeliveryMode, 1776
- persistent
  - activemq::commands::Message, 2613
- physicalName
  - activemq::commands::ActiveMQDestination, 323
- PI
  - decaf::lang::Math, 2592
- Pointer
  - decaf::lang::Pointer, 3035, 3036
- PointerType
  - decaf::lang::ArrayPointer, 739
  - decaf::lang::Pointer, 3035
- pool
  - activemq::util::concurrent::BlockingQueue, 3834
  - activemq::util::concurrent::SynchronousQueue, 3835
  - activemq::util::PriorityQueue, 3122
- PooledSession
  - activemq::util::concurrent::PooledSession, 3045
- PooledThread
  - activemq::util::concurrent::PooledThread, 3057
- pop

- decaf::util::StlQueue, 3727
- port
  - decaf::net::SocketImpl, 3644
- PortUnreachableException
  - decaf::net::PortUnreachableException, 3061, 3062
- Pos
  - deflate.h, 4623
- pos
  - gz\_state, 2041
- Posf
  - deflate.h, 4623
- position
  - decaf::nio::Buffer, 941
- POSITIVE\_INFINITY
  - decaf::lang::Double, 1853
  - decaf::lang::Float, 1973
- pow
  - decaf::lang::Math, 2585
- prefetch
  - activemq::commands::ConsumerControl, 1446
- PrefetchPolicy
  - activemq::core::PrefetchPolicy, 3064
- prefetchSize
  - activemq::commands::ConsumerInfo, 1509
- PRESET\_DICT
  - zutil.h, 4639
- prev
  - internal\_state, 2191
- prev\_length
  - internal\_state, 2191
- prev\_match
  - internal\_state, 2191
- previous
  - decaf::util::ListIterator, 2419
- previousIndex
  - decaf::util::ListIterator, 2419
- PrimitiveList
  - activemq::util::PrimitiveList, 3070
- PrimitiveMap
  - activemq::util::PrimitiveMap, 3082
- PrimitiveType
  - activemq::util::PrimitiveValueNode, 3104
- PrimitiveTypesMarshaller
  - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3092
- PrimitiveValueConverter
  - activemq::util::PrimitiveValueConverter, 3099
- PrimitiveValueNode
  - activemq::util::PrimitiveValueNode, 3104–3107
- printStackTrace
  - cms::CMSEException, 1193
  - decaf::lang::Exception, 1892
  - decaf::lang::Throwable, 3898
- priority
  - activemq::commands::ConsumerInfo, 1509
  - activemq::commands::Message, 2613
- PriorityQueue
  - decaf::util::PriorityQueue, 3118, 3119
- PriorityQueueIterator
  - decaf::util::PriorityQueue, 3124
- processBeginTransaction
  - activemq::state::CommandVisitor, 1235
  - activemq::state::CommandVisitorAdapter, 1245
  - activemq::state::ConnectionStateTracker, 1435
- processBrokerError
  - activemq::state::CommandVisitor, 1235
  - activemq::state::CommandVisitorAdapter, 1245
- processBrokerInfo
  - activemq::state::CommandVisitor, 1236
  - activemq::state::CommandVisitorAdapter, 1245
- processCommitTransactionOnePhase
  - activemq::state::CommandVisitor, 1236
  - activemq::state::CommandVisitorAdapter, 1245
  - activemq::state::ConnectionStateTracker, 1435
- processCommitTransactionTwoPhase
  - activemq::state::CommandVisitor, 1236
  - activemq::state::CommandVisitorAdapter, 1245
  - activemq::state::ConnectionStateTracker, 1436
- processConnectionControl
  - activemq::state::CommandVisitor, 1236
  - activemq::state::CommandVisitorAdapter, 1245
- processConnectionError
  - activemq::state::CommandVisitor, 1236

activemq::state::CommandVisitorAdapter,	activemq::state::ConnectionStateTracker,
1245	1436
processConnectionInfo	processMessageAck
activemq::state::CommandVisitor, 1236	activemq::state::CommandVisitor, 1238
activemq::state::CommandVisitorAdapter,	activemq::state::CommandVisitorAdapter,
1245	1245
activemq::state::ConnectionStateTracker,	activemq::state::ConnectionStateTracker,
1436	1437
processConsumerControl	processMessageDispatch
activemq::state::CommandVisitor, 1236	activemq::state::CommandVisitor, 1238
activemq::state::CommandVisitorAdapter,	activemq::state::CommandVisitorAdapter,
1245	1245
processConsumerInfo	processMessageDispatchNotification
activemq::state::CommandVisitor, 1237	activemq::state::CommandVisitor, 1239
activemq::state::CommandVisitorAdapter,	activemq::state::CommandVisitorAdapter,
1245	1245
activemq::state::ConnectionStateTracker,	processMessagePull
1436	activemq::state::CommandVisitor, 1239
processControlCommand	activemq::state::CommandVisitorAdapter,
activemq::state::CommandVisitor, 1237	1245
activemq::state::CommandVisitorAdapter,	processPrepareTransaction
1245	activemq::state::CommandVisitor, 1239
processDestinationInfo	activemq::state::CommandVisitorAdapter,
activemq::state::CommandVisitor, 1237	1245
activemq::state::CommandVisitorAdapter,	activemq::state::ConnectionStateTracker,
1245	1437
activemq::state::ConnectionStateTracker,	processProducerAck
1436	activemq::state::CommandVisitor, 1239
processEndTransaction	activemq::state::CommandVisitorAdapter,
activemq::state::CommandVisitor, 1237	1245
activemq::state::CommandVisitorAdapter,	processProducerInfo
1245	activemq::state::CommandVisitor, 1239
activemq::state::ConnectionStateTracker,	activemq::state::CommandVisitorAdapter,
1436	1245
processFlushCommand	activemq::state::ConnectionStateTracker,
activemq::state::CommandVisitor, 1237	1437
activemq::state::CommandVisitorAdapter,	processRecoverTransactions
1245	activemq::state::CommandVisitor, 1239
processForgetTransaction	activemq::state::CommandVisitorAdapter,
activemq::state::CommandVisitor, 1238	1245
activemq::state::CommandVisitorAdapter,	processRemoveConnection
1245	activemq::state::CommandVisitor, 1239
processKeepAliveInfo	activemq::state::CommandVisitorAdapter,
activemq::state::CommandVisitor, 1238	1245
activemq::state::CommandVisitorAdapter,	activemq::state::ConnectionStateTracker,
1245	1437
processMessage	processRemoveConsumer
activemq::state::CommandVisitor, 1238	activemq::state::CommandVisitor, 1239
activemq::state::CommandVisitorAdapter,	activemq::state::CommandVisitorAdapter,
1245	1245

- activemq::state::ConnectionStateTracker, 1437
- processRemoveDestination
  - activemq::state::CommandVisitor, 1240
  - activemq::state::CommandVisitorAdapter, 1245
  - activemq::state::ConnectionStateTracker, 1437
- processRemoveInfo
  - activemq::state::CommandVisitor, 1240
  - activemq::state::CommandVisitorAdapter, 1245
- processRemoveProducer
  - activemq::state::CommandVisitor, 1240
  - activemq::state::CommandVisitorAdapter, 1246
  - activemq::state::ConnectionStateTracker, 1438
- processRemoveSession
  - activemq::state::CommandVisitor, 1240
  - activemq::state::CommandVisitorAdapter, 1247
  - activemq::state::ConnectionStateTracker, 1438
- processRemoveSubscriptionInfo
  - activemq::state::CommandVisitor, 1240
  - activemq::state::CommandVisitorAdapter, 1247
- processReplayCommand
  - activemq::state::CommandVisitor, 1241
  - activemq::state::CommandVisitorAdapter, 1247
- processResponse
  - activemq::state::CommandVisitor, 1241
  - activemq::state::CommandVisitorAdapter, 1247
- processRollbackTransaction
  - activemq::state::CommandVisitor, 1241
  - activemq::state::CommandVisitorAdapter, 1247
  - activemq::state::ConnectionStateTracker, 1438
- processSessionInfo
  - activemq::state::CommandVisitor, 1241
  - activemq::state::CommandVisitorAdapter, 1247
  - activemq::state::ConnectionStateTracker, 1438
- processShutdownInfo
  - activemq::state::CommandVisitor, 1241
- activemq::state::CommandVisitorAdapter, 1247
- processTransactionInfo
  - activemq::state::CommandVisitor, 1241
  - activemq::state::CommandVisitorAdapter, 1247
- processWireFormat
  - activemq::state::CommandVisitor, 1241
  - activemq::state::CommandVisitorAdapter, 1248
- PRODUCER\_ADVISORY\_PREFIX
- activemq::commands::ActiveMQDestination, 323
- ProducerAck
  - activemq::commands::ProducerAck, 3126
- ProducerAckMarshaller
  - activemq::wireformat::openwire::marshal::v1::ProducerAckM, 3151
  - activemq::wireformat::openwire::marshal::v2::ProducerAckM, 3130
  - activemq::wireformat::openwire::marshal::v3::ProducerAckM, 3138
  - activemq::wireformat::openwire::marshal::v4::ProducerAckM, 3134
  - activemq::wireformat::openwire::marshal::v5::ProducerAckM, 3142
  - activemq::wireformat::openwire::marshal::v6::ProducerAckM, 3147
- ProducerExecutor
  - activemq::cmsutil::CmsTemplate, 1215
  - activemq::cmsutil::CmsTemplate::ProducerExecutor, 3156
- ProducerId
  - activemq::commands::ProducerId, 3158
- ProducerIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::ProducerIdMar, 3182
  - activemq::wireformat::openwire::marshal::v2::ProducerIdMar, 3162
  - activemq::wireformat::openwire::marshal::v3::ProducerIdMar, 3170
  - activemq::wireformat::openwire::marshal::v4::ProducerIdMar, 3166
  - activemq::wireformat::openwire::marshal::v5::ProducerIdMar, 3174
- activemq::commands::Message, 2613
- activemq::commands::MessageId, 2755
- activemq::commands::ProducerAck, 3128
- activemq::commands::ProducerInfo, 3190

Generated on Tue Mar 8 2011 17:10:05 for activemq-cpp-3.2.5 by Doxygen

Generated on Tue Mar 8 2011 17:10:05 for activemq-cpp-3.2.5 by Doxygen



---

decaf::internal::util::ByteArrayAdapter::readConfiguration  
     1002  
 decaf::io::InputStream, 2110–2112  
 decaf::io::Reader, 3257–3259  
 decaf::lang::Readable, 3252  
 decaf::nio::CharBuffer, 1164  
 readAsciiString  
     activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,  
         826  
 readBoolean  
     activemq::commands::ActiveMQBytesMessage, 221  
     activemq::commands::ActiveMQStreamMessage, 540  
     activemq::wireformat::openwire::utils::ReadFromStream,  
         865  
     cms::BytesMessage, 1084  
     cms::StreamMessage, 3763  
     decaf::io::DataInput, 1605  
     decaf::io::DataInputStream, 1615  
 readByte  
     activemq::commands::ActiveMQBytesMessage, 222  
     activemq::commands::ActiveMQStreamMessage, 541  
     cms::BytesMessage, 1084  
     cms::StreamMessage, 3763  
     decaf::io::DataInput, 1605  
     decaf::io::DataInputStream, 1615  
 readBytes  
     activemq::commands::ActiveMQBytesMessage, 222, 223  
     activemq::commands::ActiveMQStreamMessage, 541, 542  
     cms::BytesMessage, 1085, 1086  
     cms::StreamMessage, 3764, 3765  
 readChar  
     activemq::commands::ActiveMQBytesMessage, 224  
     activemq::commands::ActiveMQStreamMessage, 543  
     cms::BytesMessage, 1087  
     cms::StreamMessage, 3766  
     decaf::io::DataInput, 1606  
     decaf::io::DataInputStream, 1615  
 ReadChecker  
     activemq::transport::inactivity::InactivityMonitor, 2069  
     activemq::transport::inactivity::ReadChecker, 3253  
     decaf::util::logging::LogManager, 2485, 2486  
     readDouble  
     activemq::commands::ActiveMQBytesMessage, 224  
     activemq::commands::ActiveMQStreamMessage, 544  
     BaseDataStreamMarshaller,  
         826  
     cms::BytesMessage, 1087  
     cms::StreamMessage, 3766  
     decaf::io::DataInput, 1606  
     decaf::io::DataInputStream, 1616  
     decaf::io::Reader, 3255  
     activemq::commands::ActiveMQBytesMessage, 225  
     activemq::commands::ActiveMQStreamMessage, 544  
     cms::BytesMessage, 1087  
     cms::StreamMessage, 3767  
     decaf::io::DataInput, 1606  
     decaf::io::DataInputStream, 1616  
     decaf::io::DataInput, 1607, 1608  
     decaf::io::DataInputStream, 1617  
     readInt  
         activemq::commands::ActiveMQBytesMessage, 225  
         activemq::commands::ActiveMQStreamMessage,  
             544  
         cms::BytesMessage, 1088  
         cms::StreamMessage, 3767  
         decaf::io::DataInput, 1608  
         decaf::io::DataInputStream, 1618  
     readLine  
         decaf::io::DataInput, 1609  
         decaf::io::DataInputStream, 1618  
     readLock  
         decaf::util::concurrent::locks::ReadWriteLock,  
             3265  
     readLong  
         activemq::commands::ActiveMQBytesMessage, 225  
         activemq::commands::ActiveMQStreamMessage, 545  
         cms::BytesMessage, 1088  
         cms::StreamMessage, 3768  
         decaf::io::DataInput, 1609  
         decaf::io::DataInputStream, 1619

---

readOnly  
     decaf::internal::nio::CharArrayBuffer, 1148  
 readonly  
     decaf::io::FileDescriptor, 1949  
 ReadOnlyBufferException  
     decaf::nio::ReadOnlyBufferException, 3261, 3262  
 readShort  
     activemq::commands::ActiveMQBytesMessage, 1100  
     226  
     activemq::commands::ActiveMQStreamMessage, 545  
     cms::BytesMessage, 1089  
     cms::StreamMessage, 3768  
     decaf::io::DataInput, 1610  
     decaf::io::DataInputStream, 1619  
 readString  
     activemq::commands::ActiveMQBytesMessage, 226  
     activemq::commands::ActiveMQStreamMessage, 546  
     cms::BytesMessage, 1089  
     cms::StreamMessage, 3769  
     decaf::io::DataInput, 1610  
     decaf::io::DataInputStream, 1620  
 readString16  
     activemq::util::MarshallingSupport, 2573  
 readString32  
     activemq::util::MarshallingSupport, 2573  
 readUnsignedByte  
     decaf::io::DataInput, 1610  
     decaf::io::DataInputStream, 1620  
 readUnsignedShort  
     activemq::commands::ActiveMQBytesMessage, 227  
     activemq::commands::ActiveMQStreamMessage, 546  
     cms::BytesMessage, 1090  
     cms::StreamMessage, 3770  
     decaf::io::DataInput, 1611  
     decaf::io::DataInputStream, 1620  
 readUTF  
     activemq::commands::ActiveMQBytesMessage, 227  
     cms::BytesMessage, 1090  
     decaf::io::DataInput, 1611  
     decaf::io::DataInputStream, 1621  
 ready  
     decaf::io::InputStreamReader, 2118  
     decaf::io::Reader, 3259  
     rebalanceConnection  
     activemq::commands::ConnectionControl, 1306  
 RECEIPT  
     activemq::wireformat::stomp::StompCommandConstants, 3740  
 receive  
     activemq::cmsutil::CachedConsumer, 226  
     activemq::cmsutil::CmsTemplate, 1208, 1209  
     activemq::core::ActiveMQConsumer, 308  
     cms::MessageConsumer, 2676  
     RECEIVE\_TIMEOUT\_INDEFINITE\_WAIT  
     activemq::cmsutil::CmsTemplate, 1215  
     RECEIVE\_TIMEOUT\_NO\_WAIT  
     activemq::cmsutil::CmsTemplate, 1215  
     ReceiveExecutor  
     activemq::cmsutil::CmsTemplate, 1215  
     activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3266  
     receiveNoWait  
     activemq::cmsutil::CachedConsumer, 1100  
     activemq::core::ActiveMQConsumer, 309  
     cms::MessageConsumer, 2677  
     receiveSelected  
     activemq::cmsutil::CmsTemplate, 1209, 1210  
     recievedByDFBridge  
     activemq::commands::Message, 2613  
     Message  
     activemq::transport::failover::FailoverTransport, 1938  
     activemq::transport::IOTransport, 2217  
     activemq::transport::mock::MockTransport, 2860  
     activemq::transport::Transport, 4000  
     activemq::transport::TransportFilter, 4010  
 reconnectTo  
     activemq::commands::ConnectionControl, 1306  
 recover  
     activemq::cmsutil::PooledSession, 3054  
     activemq::core::ActiveMQSession, 528  
     cms::Session, 3473  
 redeliveryCounter

- activemq::commands::Message, 2613
- activemq::commands::MessageDispatch, 2683
- RedeliveryPolicy
  - activemq::core::RedeliveryPolicy, 3269
- redispatch
  - activemq::core::ActiveMQSession, 529
- ReentrantLock
  - decaf::util::concurrent::locks::ReentrantLock, 3275
- ReferenceType
  - decaf::lang::ArrayPointer, 739
  - decaf::lang::Pointer, 3035
- registerFactory
  - activemq::transport::TransportRegistry, 4017
  - activemq::wireformat::WireFormatRegistry, 4131
- rejectedExecution
  - decaf::util::concurrent::RejectedExecutionHandler, 3283
- RejectedExecutionException
  - decaf::util::concurrent::RejectedExecutionException, 3281, 3282
- relativize
  - decaf::net::URI, 4041
- release
  - decaf::lang::ArrayPointer, 743
  - decaf::lang::Pointer, 3039
  - decaf::util::concurrent::atomic::AtomicReference, 755
  - decaf::util::concurrent::Semaphore, 344
- releaseAll
  - activemq::cmsutil::ResourceLifecycleManager, 3379
- remaining
  - decaf::nio::Buffer, 941
- remainingCapacity
  - decaf::util::concurrent::BlockingQueue, 855
  - decaf::util::concurrent::SynchronousQueue, 3836
- remove
  - activemq::util::ActiveMQProperties, 478
  - cms::CMSProperties, 1198
  - decaf::internal::util::TimerTaskHeap, 3919
  - decaf::util::AbstractCollection, 168
  - decaf::util::AbstractQueue, 179
  - decaf::util::Collection, 1223
  - decaf::util::concurrent::ConcurrentMap, 1262
  - decaf::util::concurrent::ConcurrentStlMap, 1277
  - decaf::util::concurrent::SynchronousQueue, 3836
  - decaf::util::Iterator, 2223
  - decaf::util::List, 2415
  - decaf::util::Map, 2548
  - decaf::util::PriorityQueue, 3123
  - decaf::util::Properties, 3224
  - decaf::util::Queue, 3243
  - decaf::util::StlList, 3706
  - decaf::util::StlMap, 3719
  - decaf::util::StlSet, 3737
  - removeAll
    - activemq::core::MessageDispatchChannel, 2688
  - decaf::util::AbstractCollection, 169
  - decaf::util::AbstractSet, 181
  - decaf::util::Collection, 1224
  - decaf::util::concurrent::SynchronousQueue, 3837
  - removeConsumer
    - activemq::core::ActiveMQSession, 529
    - activemq::state::SessionState, 3537
  - removeDispatcher
    - activemq::core::ActiveMQConnection, 275
  - RemoveInfo
    - activemq::commands::RemoveInfo, 3285
  - RemoveInfoMarshaller
    - activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller, 3302
    - activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller, 3289
    - activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller, 3297
    - activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller, 3310
    - activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller, 3306
    - activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller, 3293
  - removeProducer
    - activemq::core::ActiveMQConnection, 275
    - activemq::core::ActiveMQSession, 529

- activemq::state::SessionState, 3537
- removeProperty
  - activemq::wireformat::stomp::StompFrame, 3744
- removePropertyChangeListener
  - decaf::util::logging::LogManager, 2486
- removeSession
  - activemq::core::ActiveMQConnection, ReplayCommand, 275
  - activemq::state::ConnectionState, 1432
- RemoveSubscriptionInfo
  - ReplayCommandMarshaller
  - activemq::commands::RemoveSubscriptionInfo, 3315
- RemoveSubscriptionInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller, 3319
  - activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller, 3328
  - activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller, 3323
  - activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller, 3340
  - activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller, 3336
  - activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller, 3332
- removeSynchronization
  - activemq::core::ActiveMQTransactionContext, request, 729
- removeTask
  - activemq::threads::CompositeTaskRunner, 1257
- removeTempDestination
  - activemq::state::ConnectionState, 1432
- RemoveTransactionAction
  - activemq::state::ConnectionStateTracker, 1439
- removeTransactionState
  - activemq::state::ConnectionState, 1432
- removeTransportListener
  - activemq::core::ActiveMQConnection, 275
- removeURI
  - activemq::transport::CompositeTransport, reserve, 1260
  - activemq::transport::failover::FailoverTransport, 1938
  - activemq::transport::failover::URIPool, reset, 4056
- renegotiateWireFormat
- activemq::wireformat::openwire::OpenWireFormat, 2979
- decaf::util::concurrent::ConcurrentMap, 1263, 1264
- decaf::util::concurrent::ConcurrentStlMap, 1278, 1279
- activemq::commands::ReplayCommand, 3345
- activemq::wireformat::openwire::marshal::v1::ReplayCommand, 3352
- activemq::wireformat::openwire::marshal::v2::ReplayCommand, 3357
- activemq::wireformat::openwire::marshal::v3::ReplayCommand, 3361
- activemq::wireformat::openwire::marshal::v4::ReplayCommand, 3348
- activemq::wireformat::openwire::marshal::v5::ReplayCommand, 3369
- activemq::wireformat::openwire::marshal::v6::ReplayCommand, 3365
- replyTo
  - activemq::commands::Message, 2613
- reportError
  - decaf::util::logging::Handler, 2045
- activemq::transport::correlator::ResponseCorrelator, 3387
- activemq::transport::failover::FailoverTransport, 1938, 1939
- activemq::transport::IOTransport, 2218
- activemq::transport::logging::LoggingTransport, 2479
- activemq::transport::mock::MockTransport, 2861
- activemq::transport::Transport, 4000, 4001
- activemq::transport::TransportFilter, 4011
- activemq::wireformat::openwire::OpenWireFormatNegotiator, 2988
- decaf::util::concurrent::ThreadPool, 3893
- z\_stream\_s, 4175
- activemq::commands::ActiveMQBytesMessage, 228

- activemq::commands::ActiveMQStreamMessage, 3380
- 547
- ResponseCorrelator
- activemq::state::ConnectionState, 1432
- activemq::transport::correlator::ResponseCorrelator, 3385
- cms::BytesMessage, 1091
- decaf::internal::util::TimerTaskHeap, 3919
- ResponseMarshaller
- activemq::wireformat::openwire::marshal::v1::ResponseMarshaller, 3410
- decaf::io::BufferedInputStream, 947
- activemq::wireformat::openwire::marshal::v2::ResponseMarshaller, 3395
- decaf::io::ByteArrayInputStream, 1044
- activemq::wireformat::openwire::marshal::v3::ResponseMarshaller, 3405
- decaf::io::ByteArrayOutputStream, 1048
- activemq::wireformat::openwire::marshal::v4::ResponseMarshaller, 3390
- decaf::io::FilterInputStream, 1955
- activemq::wireformat::openwire::marshal::v5::ResponseMarshaller, 3400
- decaf::io::InputStream, 2113
- activemq::wireformat::openwire::marshal::v6::ResponseMarshaller, 3415
- decaf::io::PushbackInputStream, 3236
- decaf::io::Reader, 3260
- decaf::lang::ArrayPointer, 743
- decaf::lang::Pointer, 3039
- decaf::nio::Buffer, 941
- decaf::util::logging::LogManager, 2486
- decaf::util::StringTokenizer, 3781
- restore
- decaf::util::zip::Adler32, 731
- activemq::state::ConnectionStateTracker, 1438
- decaf::util::zip::Checksum, 1175
- restoreTransport
- decaf::util::zip::CRC32, 1569
- activemq::transport::failover::FailoverTransport, 1939
- decaf::util::zip::Deflater, 1764
- decaf::util::zip::Inflater, 2093
- decaf::util::zip::InflaterInputStream, 2103
- result
- resize
- activemq::commands::IntegerResponse, 2162
- decaf::internal::util::ByteArrayAdapter, 1002
- resume
- resolve
- activemq::commands::ConnectionControl, 1306
- decaf::net::URI, 4042
- retainAll
- resolveDestinationName
- decaf::util::AbstractCollection, 169
- activemq::cmsutil::CmsDestinationAccessor, 1190
- decaf::util::Collection, 1225
- activemq::cmsutil::DestinationResolver, 1811
- decaf::util::concurrent::SynchronousQueue, 3837
- activemq::cmsutil::DynamicDestinationResolver, 1880
- activemq::commands::ConsumerInfo, 1509
- ResolveProducerExecutor
- returnInstance
- activemq::cmsutil::CmsTemplate, 1215
- decaf::util::logging::LogWriter, 2494
- activemq::cmsutil::CmsTemplate::ResolveProducer, 3373
- returnSession
- ResolveReceiveExecutor
- activemq::cmsutil::SessionPool, 3535
- activemq::cmsutil::CmsTemplate, 1215
- reverse
- activemq::cmsutil::CmsTemplate::ResolveReceive, 3374
- ReceiveHangupInteger, 2154
- ResourceLifecycleManager
- decaf::lang::Long, 2505
- decaf::util::StlQueue, 3728
- activemq::cmsutil::ResourceLifecycleManager, 3377
- decaf::lang::Integer, 2154
- decaf::internal::util::ResourceLifecycleManager, 3376
- decaf::lang::Long, 2505
- decaf::lang::Short, 3546
- Response
- rewind

- decaf::nio::Buffer, 942
- rollback
  - activemq::cmsutil::PooledSession, 3055
  - activemq::core::ActiveMQConsumer, 309
  - activemq::core::ActiveMQSession, 530
  - activemq::core::ActiveMQTransactionContext, 729
  - cms::Session, 3474
- rotateLeft
  - decaf::lang::Integer, 2154
  - decaf::lang::Long, 2506
- rotateRight
  - decaf::lang::Integer, 2155
  - decaf::lang::Long, 2506
- round
  - decaf::lang::Math, 2586
- run
  - activemq::threads::CompositeTaskRunner, 1258
  - activemq::threads::DedicatedTaskRunner, 1725
  - activemq::transport::inactivity::ReadChecker, 3253
  - activemq::transport::inactivity::WriteChecker, 4133
  - activemq::transport::IOTransport, 2219
  - activemq::transport::mock::InternalCommandListener, 2193
  - decaf::lang::Runnable, 3419
  - decaf::lang::Thread, 3884
  - decaf::util::concurrent::PooledThread, 3057
- RUNNABLE
  - decaf::lang::Thread, 3880
- RuntimeException
  - decaf::lang::exceptions::RuntimeException, 3422, 3423
- sane
  - inflate\_state, 2087
- schedule
  - decaf::util::Timer, 3904–3910
- scheduleAtFixedRate
  - decaf::util::Timer, 3910–3913
- scheduledExecutionTime
  - decaf::util::TimerTask, 3916
- SECONDS
  - decaf::util::concurrent::TimeUnit, 3929
- SecureRandom
  - decaf::security::SecureRandom, 3426
  - SecureRandomImpl
    - decaf::internal::security::SecureRandomImpl, 3430
    - SecureRandomSpi
      - decaf::security::SecureRandomSpi, 3433
  - gz\_state, 2041
  - SEEK\_CUR
    - zconf.h, 4632
  - SEEK\_END
    - zconf.h, 4632
  - SEEK\_SET
    - zconf.h, 4632
  - selector
    - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3267
    - activemq::commands::ConsumerInfo, 1509
    - activemq::commands::SubscriptionInfo, 3786
    - Semaphore
      - decaf::util::concurrent::Semaphore, 3437
    - semaphore
      - decaf::util::concurrent::ConditionHandle, 1291
  - SEND
    - activemq::wireformat::stomp::StompCommandConstants, 3740
  - send
    - activemq::cmsutil::CachedProducer, 1105, 1106
    - activemq::cmsutil::CmsTemplate, 1210, 1211
    - activemq::core::ActiveMQProducer, 471, 472
    - activemq::core::ActiveMQSession, 530
    - cms::MessageProducer, 2813–2815
  - SendExecutor
    - activemq::cmsutil::CmsTemplate, 1215
    - activemq::cmsutil::CmsTemplate::SendExecutor, 3446
  - sendPullRequest
    - activemq::core::ActiveMQConnection, 276
  - sendUrgentData
    - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2951
    - decaf::net::Socket, 3621
    - decaf::net::SocketImpl, 3642

ServerSocket	decaf::util::concurrent::atomic::AtomicBoolean,
decaf::net::ServerSocket, 3449–3451	747
decaf::net::Socket, 3625	decaf::util::concurrent::atomic::AtomicInteger,
ServerSocketFactory	753
decaf::net::ServerSocketFactory, 3458	decaf::util::concurrent::atomic::AtomicReference,
serviceName	758
activemq::commands::DiscoveryEvent,	decaf::util::List, 2416
1815	decaf::util::ListIterator, 2419
SESSION_TRANSACTED	decaf::util::StlList, 3707
cms::Session, 3464	setAbsolute
SessionId	decaf::internal::net::URIType, 4068
activemq::commands::SessionId, 3478	setAckHandler
sessionId	activemq::commands::Message, 2608
activemq::commands::ConsumerId, 1476	setAckMode
activemq::commands::ProducerId, 3161	activemq::commands::SessionInfo, 3507
activemq::commands::SessionInfo, 3508	setAckType
SessionIdMarshaller	activemq::commands::MessageAck, 2646
activemq::wireformat::openwire::marshaller::ActiveSessionIdMarshaller,	setAddSessionIdMatch
3502	activemq::commands::ConsumerInfo,
activemq::wireformat::openwire::marshaller::v2::SessionIdMarshaller,	3507
3482	setAdvisory
activemq::wireformat::openwire::marshaller::v3::SessionIdMarshaller,	ActiveMQDestination,
3498	320
activemq::wireformat::openwire::marshaller::v4::SessionIdMarshaller,	setAllowSessionIdMatch
3486	activemq::core::ActiveMQConnection,
activemq::wireformat::openwire::marshaller::v5::SessionIdMarshaller,	3516
3494	activemq::core::ActiveMQConnectionFactory,
activemq::wireformat::openwire::marshaller::v6::SessionIdMarshaller,	3489
3490	setArrival
SessionInfo	activemq::commands::Message, 2609
activemq::commands::SessionInfo, 3506	setAuthority
SessionInfoMarshaller	decaf::internal::net::URIType, 4069
activemq::wireformat::openwire::marshaller::ActiveSessionInfoMarshaller,	setBackQoSNonInflimit
3518	activemq::core::policies::DefaultRedeliveryPolicy,
activemq::wireformat::openwire::marshaller::v2::SessionInfoMarshaller,	3533
3527	activemq::core::RedeliveryPolicy, 3271
activemq::wireformat::openwire::marshaller::v3::SessionInfoMarshaller,	setSessionInfoMarshaller
3522	activemq::transport::failover::FailoverTransport,
3531	1940
activemq::wireformat::openwire::marshaller::v4::SessionInfoMarshaller,	setBackQoS
3514	activemq::transport::failover::FailoverTransport,
activemq::wireformat::openwire::marshaller::v5::SessionInfoMarshaller,	1940
3510	setBackupPoolSize
activemq::wireformat::openwire::marshaller::v6::SessionInfoMarshaller,	activemq::transport::failover::BackupTransportPool,
3510	764
SessionPool	activemq::transport::failover::FailoverTransport,
activemq::cmsutil::SessionPool, 3535	1940
SessionState	setBlockSize
activemq::state::SessionState, 3537	decaf::util::concurrent::ThreadPool, 3894
set	setBody

activemq::wireformat::stomp::StompFrame, activemq::commands::BrokerInfo, 908  
 3745  
 activemq::core::ActiveMQConnection,  
 setBodyBytes 276  
 activemq::commands::ActiveMQBytesMessage, activemq::core::ActiveMQConnectionFactory,  
 228 289  
 cms::BytesMessage, 1091  
 setBrowser  
 setBool  
 activemq::util::PrimitiveList, 3075  
 activemq::util::PrimitiveMap, 3087  
 activemq::util::PrimitiveValueNode, 3111  
 setBoolean  
 activemq::commands::ActiveMQMapMessage, activemq::util::PrimitiveList, 3075  
 360  
 activemq::util::PrimitiveMap, 3087  
 cms::MapMessage, 2558  
 activemq::util::PrimitiveValueNode, 3112  
 cms::MapMessage, 2559  
 setBooleanProperty  
 activemq::commands::ActiveMQMessage, setByteArray  
 431  
 activemq::util::PrimitiveList, 3076  
 activemq::wireformat::openwire::utils::MessagePropertyInterceptor, activemq::util::PrimitiveMap, 3087  
 2822  
 activemq::util::PrimitiveValueNode, 3112  
 cms::Message, 2632  
 decaf::io::BlockingByteArrayInputStream,  
 setBranchQualifier 847  
 activemq::commands::XATransactionId, decaf::io::ByteArrayInputStream, 1045  
 4146  
 setByteProperty  
 setBrokerId  
 activemq::commands::BrokerInfo, 908  
 432  
 setBrokerInTime  
 activemq::commands::Message, 2609  
 2822  
 setBrokerMasterConnector  
 cms::Message, 2633  
 activemq::commands::ConnectionInfo, setBytes  
 1398  
 activemq::commands::ActiveMQMapMessage,  
 setBrokerName 361  
 activemq::commands::BrokerInfo, 908  
 cms::MapMessage, 2559  
 activemq::commands::DiscoveryEvent, setCacheEnabled  
 1814  
 activemq::commands::WireFormatInfo,  
 setBrokerOutTime 4101  
 activemq::commands::Message, 2609  
 activemq::wireformat::openwire::OpenWireFormat,  
 setBrokerPath 2980  
 activemq::commands::ConnectionInfo, setCacheSize  
 1398  
 activemq::commands::WireFormatInfo,  
 activemq::commands::ConsumerInfo, 4101  
 1507  
 activemq::wireformat::openwire::OpenWireFormat,  
 activemq::commands::DestinationInfo, 2980  
 1783  
 setCause  
 activemq::commands::Message, 2609  
 activemq::commands::BrokerError, 871  
 activemq::commands::ProducerInfo, 3188  
 setBrokerSequenceId  
 activemq::commands::ActiveMQMapMessage,  
 activemq::commands::MessageId, 2754 361  
 setBrokerUploadUrl  
 activemq::util::PrimitiveList, 3076  
 activemq::commands::BrokerInfo, 908  
 activemq::util::PrimitiveMap, 3088  
 setBrokerURL  
 activemq::util::PrimitiveValueNode, 3112



---

cms::MapMessage, 2560  
 setCipherSuites  
   decaf::net::ssl::SSLParameters, 3660  
 setClientID  
   activemq::core::ActiveMQConnection, 276  
   cms::Connection, 1300  
 setClientId  
   activemq::commands::ConnectionInfo, 1398  
   activemq::commands::JournalTopicAck, 2255  
   activemq::commands::RemoveSubscriptionInfo, 3316  
   activemq::commands::SubscriptionInfo, 3785  
   activemq::core::ActiveMQConnectionFactory, 290  
 setClientMaster  
   activemq::commands::ConnectionInfo, 1398  
 setClose  
   activemq::commands::ConnectionControl, 1305  
   activemq::commands::ConsumerControl, 1445  
 setClosed  
   activemq::transport::failover::BackupTransport, 761  
 setCloseTimeout  
   activemq::core::ActiveMQConnection, 277  
   activemq::core::ActiveMQConnectionFactory, 290  
 setCluster  
   activemq::commands::Message, 2609  
 setCMSCorrelationID  
   activemq::commands::ActiveMQMessageTemplate, 432  
   cms::Message, 2633  
 setCMSDeliveryMode  
   activemq::commands::ActiveMQMessageTemplate, 432  
   cms::Message, 2634  
 setCMSDestination  
   activemq::commands::ActiveMQMessageTemplate, 433  
   cms::Message, 2635  
 setCMSExpiration  
   activemq::commands::ActiveMQMessageTemplate, 433  
   cms::Message, 2635  
   activemq::commands::ActiveMQMessageTemplate, 433  
   cms::Message, 2636  
   setCMSPriority  
   activemq::commands::ActiveMQMessageTemplate, 433  
   cms::Message, 2636  
   setCMSRedelivered  
   activemq::commands::ActiveMQMessageTemplate, 434  
   cms::Message, 2637  
   setCMSReplyTo  
   activemq::commands::ActiveMQMessageTemplate, 434  
   cms::Message, 2637  
   setCMSTimestamp  
   activemq::commands::ActiveMQMessageTemplate, 434  
   cms::Message, 2638  
   setCMSType  
   activemq::commands::ActiveMQMessageTemplate, 435  
   cms::Message, 2638  
   setCollisionAvoidancePercent  
   activemq::core::policies::DefaultRedeliveryPolicy, 1733  
   activemq::core::RedeliveryPolicy, 3271  
   setCommand  
   activemq::commands::ControlCommand, 1538  
   activemq::wireformat::stomp::StompFrame, 3745  
   setCommandId  
   activemq::commands::BaseCommand, 1570  
   activemq::commands::Command, 1231  
   activemq::commands::PartialCommand, 3004  
   setComponents  
   activemq::util::CompositeData, 1254  
   setCompressed  
   activemq::commands::Message, 2609  
   setCompleteBrokers  
   activemq::commands::ConnectionControl, 1305  
   setConnection

---

activemq::commands::ActiveMQTempSetDataStructure  
 581  
 activemq::commands::Message, 2609  
 setConnectionFactory  
 activemq::cmsutil::CmsAccessor, 1187  
 setConnectionId  
 activemq::commands::BrokerInfo, 908  
 activemq::commands::ConnectionError  
 1335  
 activemq::commands::ConnectionInfo, setDefaultDestinationName  
 1398  
 activemq::commands::ConsumerId, 1476  
 activemq::commands::DestinationInfo, 1783  
 activemq::commands::LocalTransactionId  
 2424  
 activemq::commands::ProducerId, 3160  
 activemq::commands::RemoveSubscriptionInfo  
 3317  
 activemq::commands::SessionId, 3480  
 activemq::commands::TransactionInfo, 3962  
 setConnectionInterruptProcessingComplete  
 activemq::state::ConnectionState, 1432  
 activemq::transport::failover::FailoverTransport  
 1940  
 setConsumerId  
 activemq::commands::ConsumerControl, 1445  
 activemq::commands::ConsumerInfo, 1507  
 activemq::commands::MessageAck, 2647  
 activemq::commands::MessageDispatch, 2681  
 activemq::commands::MessageDispatchNotification, 2719  
 activemq::commands::MessagePull, 2827  
 setContent  
 activemq::commands::Message, 2609  
 setCorrelationId  
 activemq::commands::Message, 2610  
 activemq::commands::MessagePull, 2827  
 activemq::commands::Response, 3381  
 setData  
 activemq::commands::DataArrayResponse, 1574  
 activemq::commands::DataResponse, 1634  
 activemq::commands::PartialCommand, 3005  
 activemq::commands::Message, 2610  
 setDefault  
 decaf::net::ssl::SSLContext, 3655  
 setDefaultClientId  
 activemq::core::ActiveMQConnection, 277  
 setDefaultDestination  
 activemq::cmsutil::CmsTemplate, 1212  
 setDefaultDestinationName  
 activemq::cmsutil::CmsTemplate, 1212  
 setDeletedByBroker  
 activemq::commands::ActiveMQBlobMessage, 188  
 setDeliveryMode  
 activemq::cmsutil::CachedProducer, 1107  
 activemq::cmsutil::CmsTemplate, 1212  
 activemq::core::ActiveMQProducer, 473  
 cms::MessageProducer, 2816  
 setDeliveryPersistent  
 activemq::cmsutil::CmsTemplate, 1212  
 setDeliverySequenceId  
 activemq::commands::MessageDispatchNotification, 2719  
 setDestination  
 activemq::commands::ConsumerControl, 1445  
 activemq::commands::ConsumerInfo, 1507  
 activemq::commands::DestinationInfo, 1783  
 activemq::commands::JournalQueueAck, 2655  
 activemq::commands::JournalTopicAck, 2655  
 activemq::commands::Message, 2610  
 activemq::commands::MessageAck, 2647  
 activemq::commands::MessageDispatch, 2682  
 activemq::commands::MessageDispatchNotification, 2719  
 activemq::commands::MessagePull, 2827  
 activemq::commands::ProducerInfo, 3189  
 activemq::commands::SubscriptionInfo, 3785  
 setDestinationResolver  
 activemq::cmsutil::CmsDestinationAccessor, 1190  
 setDictionary  
 decaf::util::zip::Deflater, 1765, 1766

- decaf::util::zip::Inflater, 2094, 2095
- setDisableMessageID
  - activemq::cmsutil::CachedProducer, 1107
  - activemq::core::ActiveMQProducer, 473
  - cms::MessageProducer, 2816
- setDisableMessageTimeStamp
  - activemq::cmsutil::CachedProducer, 1107
  - activemq::core::ActiveMQProducer, 473
  - cms::MessageProducer, 2816
- setDispatchAsync
  - activemq::commands::ConsumerInfo, 1507
  - activemq::commands::ProducerInfo, 3189
  - activemq::core::ActiveMQConnection, 277
  - activemq::core::ActiveMQConnectionFactory, 290
- setDouble
  - activemq::commands::ActiveMQMapMessage, 362
  - activemq::util::PrimitiveList, 3076
  - activemq::util::PrimitiveMap, 3088
  - activemq::util::PrimitiveValueNode, 3112
  - cms::MapMessage, 2560
- setDoubleProperty
  - activemq::commands::ActiveMQMessageTemplate, 435
  - activemq::wireformat::openwire::utils::MessagePropertyInterceptors, 2822
  - cms::Message, 2639
- setDroppable
  - activemq::commands::Message, 2610
- setDuplexConnection
  - activemq::commands::BrokerInfo, 908
- setDurableTopicPrefetch
  - activemq::core::policies::DefaultPrefetchPolicy, 1729
  - activemq::core::PrefetchPolicy, 3066
- setEnabled
  - activemq::transport::failover::BackupTransportPool, 764
- setEnabledCipherSuites
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2929
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2934
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2951
  - decaf::net::ssl::SSLServerSocket, 3666
  - decaf::net::ssl::SSLSocket, 3676
- setEnabledProtocols
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2929
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2934
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2951
  - decaf::net::ssl::SSLServerSocket, 3667
  - decaf::net::ssl::SSLSocket, 3677
- setenv
  - decaf::lang::System, 3845
- setExceptionHandler
  - activemq::commands::BrokerError, 872
- setExceptionListener
  - activemq::core::ActiveMQConnection, 278
  - activemq::core::ActiveMQConnectionFactory, 290
  - activemq::core::ActiveMQConnection, 1300
- setExclusive
  - activemq::core::ActiveMQDestination, 321
  - activemq::commands::ConsumerInfo, 1507
- setExit
  - activemq::commands::ConnectionControl, 1305
- setExpiration
  - activemq::commands::Message, 2610
- setExplicitQosEnabled
  - activemq::cmsutil::CmsTemplate, 1213
- setFailOnClose
  - activemq::transport::mock::MockTransport, 2862
- setFailOnKeepAliveSends
  - activemq::transport::mock::MockTransport, 2862
- setFailOnSocketException
  - activemq::transport::mock::MockTransport, 2862
- setFailOnSendMessage
  - activemq::transport::mock::MockTransport, 2862

- setFailOnStart
  - activemq::transport::mock::MockTransport, 2862
- setFailOnStop
  - activemq::transport::mock::MockTransport, 2862
- setFaultTolerant
  - activemq::commands::ConnectionControl, 1305
  - activemq::commands::ConnectionInfo, 1398
- setFaultTolerantConfiguration
  - activemq::commands::BrokerInfo, 908
- setFilter
  - decaf::util::logging::Handler, 2045
  - decaf::util::logging::Logger, 2472
- setFirstMessageId
  - activemq::commands::MessageAck, 2647
- setFirstNakNumber
  - activemq::commands::ReplayCommand, 3346
- setFloat
  - activemq::commands::ActiveMQMapMessage, 362
  - activemq::util::PrimitiveList, 3077
  - activemq::util::PrimitiveMap, 3088
  - activemq::util::PrimitiveValueNode, 3112
  - cms::MapMessage, 2561
- setFloatProperty
  - activemq::commands::ActiveMQMessageTemplate, 435
  - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2822
  - cms::Message, 2640
- setFlush
  - activemq::commands::ConsumerControl, 1445
- setFormatId
  - activemq::commands::XATransactionId, 4146
- setFormatter
  - decaf::util::logging::Handler, 2045
- setFragment
  - activemq::util::CompositeData, 1254
  - decaf::internal::net::URIType, 4069
- setGlobalTransactionId
  - activemq::commands::XATransactionId, 4146
- setGroupID
  - activemq::commands::Message, 2610
- setGroupSequence
  - activemq::commands::Message, 2610
- setHost
  - activemq::util::CompositeData, 1254
- setInitialDelayTime
  - activemq::transport::inactivity::InactivityMonitor, 2068
- setInitialized
  - activemq::transport::failover::FailoverTransport, 1940
- setInitialReconnectDelay
  - activemq::transport::failover::FailoverTransport, 1940
- setInitialRedeliveryDelay
  - activemq::core::policies::DefaultRedeliveryPolicy, 1733
  - activemq::core::RedeliveryPolicy, 3271
- setInput
  - decaf::util::zip::Deflater, 1766, 1767
  - decaf::util::zip::Inflater, 2095, 2096
- setInputStream
  - activemq::transport::IOTransport, 2219
- setInt
  - activemq::commands::ActiveMQMapMessage, 362
  - activemq::util::PrimitiveList, 3077
  - activemq::util::PrimitiveMap, 3088
  - activemq::util::PrimitiveValueNode, 3113
  - cms::MapMessage, 2561
- setIntProperty
  - activemq::commands::ActiveMQMessageTemplate, 436
- setKeepAlive
  - decaf::net::Socket, 3621
- setKeepAliveResponseRequired
  - activemq::transport::inactivity::InactivityMonitor, 2069
- setLastDeliveredSequenceId
  - activemq::commands::RemoveInfo, 3286
  - activemq::core::ActiveMQConsumer, 309
  - activemq::core::ActiveMQSession, 530
- setLastMessageId
  - activemq::commands::MessageAck, 2647
- setLastNakNumber
  - activemq::commands::MessageAck, 2647

- activemq::commands::ReplayCommand
  - 3346
- activemq::commands::ReplayCommandMasterBroker
  - activemq::commands::BrokerInfo, 908
- setLevel
  - decaf::util::logging::Handler, 2046
  - decaf::util::logging::Logger, 2472
  - decaf::util::logging::LogRecord, 2491
  - decaf::util::zip::Deflater, 1767
- setLimit
  - activemq::util::MemoryUsage, 2595
- setList
  - activemq::util::PrimitiveValueNode, 3113
- setLoggerName
  - decaf::util::logging::LogRecord, 2491
- setLong
  - activemq::commands::ActiveMQMapMessage
    - 363
  - activemq::util::PrimitiveList, 3078
  - activemq::util::PrimitiveMap, 3089
  - activemq::util::PrimitiveValueNode, 3113
  - cms::MapMessage, 2561
- setLongProperty
  - activemq::commands::ActiveMQMessageTemplate
    - 436
  - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2823
  - cms::Message, 2641
- setMagic
  - activemq::commands::WireFormatInfo, 4101
- setManageable
  - activemq::commands::ConnectionInfo, 1398
- setManaged
  - decaf::internal::util::GenericResource, 2038
- setMap
  - activemq::util::PrimitiveValueNode, 3113
- setMark
  - cms::CMSEException, 1193
  - decaf::lang::Exception, 1892
  - decaf::lang::Throwable, 3898
- setMarshaledForm
  - activemq::commands::BaseDataStructure, 840
  - activemq::wireformat::MarshalAware, 2566
- setMarshaledProperties
  - activemq::commands::Message, 2610
  - activemq::commands::WireFormatInfo, 4101
- setMasterBroker
  - activemq::commands::BrokerInfo, 908
- setMaxCacheSize
  - activemq::state::ConnectionStateTracker, 1439
- activemq::transport::failover::FailoverTransport, 1941
- setMaximumPendingMessageLimit
  - activemq::commands::ConsumerInfo, 1507
- setMaximumRedeliveries
  - activemq::core::policies::DefaultRedeliveryPolicy, 1734
  - activemq::core::RedeliveryPolicy, 3272
- setMaxInactivityDuration
  - activemq::commands::WireFormatInfo, 4102
  - activemq::wireformat::openwire::OpenWireFormat, 2980
- setMaxInactivityDurationInitialDelay
  - activemq::commands::WireFormatInfo, 4102
  - activemq::wireformat::openwire::OpenWireFormat, 2980
- setMaxInactivityDurationInitialDelay
  - activemq::wireformat::openwire::OpenWireFormat, 2980
- setMaxReconnectAttempts
  - activemq::transport::failover::FailoverTransport, 1941
- setMaxReconnectDelay
  - activemq::transport::failover::FailoverTransport, 1941
- setMaxThreads
  - decaf::util::concurrent::ThreadPool, 3894
- setMessage
  - activemq::commands::BrokerError, 872
  - activemq::commands::JournalTrace, 2283
  - activemq::commands::MessageDispatch, 2682
  - decaf::lang::Exception, 1892
  - decaf::util::logging::LogRecord, 2491
- setMessageAck
  - activemq::commands::JournalQueueAck, 2227
- setMessageCount
  - activemq::commands::MessageAck, 2647
- setMessageId
  - activemq::commands::JournalTopicAck, 2255
  - activemq::commands::Message, 2610

activemq::commands::MessageDispatchNotification, 2719  
 activemq::commands::MessagePull, 2827  
 setMessageIdEnabled  
   activemq::cmsutil::CmsTemplate, 1213  
 setMessageListener  
   activemq::cmsutil::CachedConsumer, 1101  
   activemq::core::ActiveMQConsumer, 309  
   cms::MessageConsumer, 2677  
 setMessageSequenceId  
   activemq::commands::JournalTopicAck, 2255  
 setMessageTimestampEnabled  
   activemq::cmsutil::CmsTemplate, 1213  
 setMimeType  
   activemq::commands::ActiveMQBlobMessage, 188  
 setName  
   activemq::commands::ActiveMQBlobMessage, 188  
   decaf::lang::Thread, 3884  
 setNeedClientAuth  
   decaf::internal::net::ssl::openssl::OpenSSLParameters, 2929  
   decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2935  
   decaf::internal::net::ssl::openssl::OpenSSLSocket, 2952  
   decaf::net::ssl::SSLParameters, 3661  
   decaf::net::ssl::SSLServerSocket, 3667  
   decaf::net::ssl::SSLSocket, 3677  
 setNetworkBrokerId  
   activemq::commands::NetworkBridgeFilter, 2880  
 setNetworkConnection  
   activemq::commands::BrokerInfo, 908  
 setNetworkConsumerPath  
   activemq::commands::ConsumerInfo, 1507  
 setNetworkProperties  
   activemq::commands::BrokerInfo, 908  
 setNetworkSubscription  
   activemq::commands::ConsumerInfo, 1507  
 setNetworkTTL  
   activemq::commands::NetworkBridgeFilter, 2880  
 setNoLocal  
   activemq::cmsutil::CmsTemplate, 1213  
   activemq::commands::ConsumerInfo, 1507  
   setNoRangeAcks  
   activemq::transport::mock::MockTransport, 2862  
   setNumReceivedMessageBeforeFail  
   activemq::transport::mock::MockTransport, 2862  
   setNumReceivedMessages  
   activemq::transport::mock::MockTransport, 2862  
   setNumSentKeepAlives  
   activemq::transport::mock::MockTransport, 2862  
   setNumSentKeepAlivesBeforeFail  
   activemq::transport::mock::MockTransport, 2862  
   setNumSentMessageBeforeFail  
   activemq::transport::mock::MockTransport, 2862  
   setNumSentMessages  
   activemq::transport::mock::MockTransport, 2862  
   decaf::internal::net::ssl::openssl::OpenSSLParameters, 2929  
   decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2935  
   decaf::internal::net::ssl::openssl::OpenSSLSocket, 2952  
   decaf::net::Socket, 3621  
   setOpaque  
   decaf::internal::net::URIType, 4069  
   setOperationType  
   activemq::commands::DestinationInfo, 1783  
   setOptimizedAcknowledge  
   activemq::commands::ConsumerInfo, 1507  
   setOption  
   decaf::internal::net::tcp::TcpSocket, 3858  
   decaf::net::SocketImpl, 3642  
   setOrdered  
   activemq::commands::ActiveMQDestination, 321  
   setOrderedTarget  
   activemq::commands::ActiveMQDestination, 321  
   setOriginalDestination  
   activemq::commands::Message, 2610  
   setOriginalTransactionId

activemq::commands::Message, 2610  
 setOutputStream  
   decaf::util::logging::StreamHandler, 3750  
 setOutgoingListener  
   activemq::transport::mock::MockTransport, 2862  
 setOutputStream  
   activemq::transport::IOTransport, 2219  
 setParameters  
   activemq::util::CompositeData, 1254  
 setParent  
   decaf::util::logging::Logger, 2472  
 setPassword  
   activemq::commands::ConnectionInfo, 1398  
   activemq::core::ActiveMQConnection, 278  
   activemq::core::ActiveMQConnectionFactory, 291  
 setPath  
   activemq::util::CompositeData, 1254  
   decaf::internal::net::URIType, 4069  
 setPeerBrokerInfos  
   activemq::commands::BrokerInfo, 908  
 setPersistent  
   activemq::commands::Message, 2610  
 setPhysicalName  
   activemq::commands::ActiveMQDestination, 321  
 setPooledThreadListener  
   decaf::util::concurrent::PooledThread, 3057  
 setPort  
   decaf::internal::net::URIType, 4070  
 setPreferredWireFormatInfo  
   activemq::wireformat::openwire::OpenWireFormat, 2981  
 setPrefetch  
   activemq::commands::ConsumerControl, 1445  
 setPrefetchPolicy  
   activemq::core::ActiveMQConnection, 278  
   activemq::core::ActiveMQConnectionFactory, 291  
 setPrefetchSize  
   activemq::commands::ConsumerInfo, 1507  
 setPrepared  
   activemq::state::TransactionState, 3991  
   setPreparedResult  
     activemq::state::TransactionState, 3991  
   setPriority  
     activemq::cmsutil::CachedProducer, 1108  
     activemq::cmsutil::CmsTemplate, 1213  
     activemq::commands::ConsumerInfo, 1507  
     activemq::commands::Message, 2610  
     activemq::core::ActiveMQProducer, 473  
     cms::MessageProducer, 2817  
     decaf::lang::Thread, 3884  
   setProducerId  
     activemq::commands::Message, 2610  
     activemq::commands::MessageId, 2754  
     activemq::commands::ProducerAck, 3127  
     activemq::commands::ProducerInfo, 3189  
   setProducerSequenceId  
     activemq::commands::MessageId, 2754  
   setProducerSessionKey  
     activemq::commands::ProducerId, 3160  
   setProducerWindowSize  
     activemq::core::ActiveMQConnection, 278  
     activemq::core::ActiveMQConnectionFactory, 291  
   setProperties  
     activemq::commands::WireFormatInfo, 4102  
     activemq::util::ActiveMQProperties, 478  
     decaf::util::logging::LogManager, 2487  
   setProperty  
     activemq::util::ActiveMQProperties, 478  
     activemq::wireformat::stomp::StompFrame, 3745  
     cms::CMSProperties, 1199  
     decaf::lang::System, 3845  
     decaf::util::Properties, 3224  
   setProtocols  
     decaf::net::ssl::SSLParameters, 3661  
   setPubSubDomain  
     activemq::cmsutil::CmsDestinationAccessor, 1190  
     activemq::cmsutil::CmsTemplate, 1213  
   setQueue  
     decaf::internal::net::URIType, 4070  
   setQueueBrowserPrefetch  
     activemq::core::policies::DefaultPrefetchPolicy, 1729  
     activemq::core::PrefetchPolicy, 3066  
   setQueuePrefetch

activemq::core::policies::DefaultPrefetchPolicy, 1729  
 activemq::core::PrefetchPolicy, 3066  
 setRandomize  
 activemq::transport::failover::FailoverTransport, 1941  
 activemq::transport::failover::URIPool, setRemoteBlobUrl, 4056  
 setReadCheckTime  
 activemq::transport::inactivity::InactivityLimit, 2069  
 setReadOnly  
 decaf::internal::nio::ByteBuffer, 1037  
 decaf::internal::nio::CharArrayBuffer, 1146  
 decaf::internal::nio::DoubleArrayBuffer, 1864  
 decaf::internal::nio::FloatArrayBuffer, 1984  
 decaf::internal::nio::IntArrayBuffer, 2129  
 decaf::internal::nio::LongArrayBuffer, 2521  
 decaf::internal::nio::ShortArrayBuffer, 3559  
 setReadOnlyBody  
 activemq::commands::Message, 2610  
 setReadOnlyProperties  
 activemq::commands::Message, 2611  
 setRebalanceConnection  
 activemq::commands::ConnectionControl, 1305  
 setReceiveBufferSize  
 decaf::net::ServerSocket, 3455  
 decaf::net::Socket, 3622  
 setReceiveTimeout  
 activemq::cmsutil::CmsTemplate, 1214  
 setRecievedByDFBridge  
 activemq::commands::Message, 2611  
 setReconnectDelay  
 activemq::transport::failover::FailoverTransport, 1305  
 1941  
 setReconnectTo  
 activemq::commands::ConnectionControl, 1305  
 setRedeliveryCounter  
 activemq::commands::Message, 2611  
 activemq::commands::MessageDispatcher, 2682  
 setRedeliveryPolicy  
 activemq::core::ActiveMQConnection, 279  
 activemq::core::ActiveMQConnectionFactory, 291  
 activemq::core::ActiveMQConsumer, 310  
 activemq::commands::ActiveMQBlobMessage, 188  
 activemq::commands::Message, 2611  
 setResponse  
 activemq::transport::correlator::FutureResponse, 2034  
 setResponseBuilder  
 activemq::transport::mock::InternalCommandListener, 2193  
 activemq::transport::mock::MockTransport, 2863  
 setResponseRequired  
 activemq::commands::BaseCommand, 770  
 activemq::commands::Command, 1231  
 setRestoreConsumers  
 activemq::state::ConnectionStateTracker, 1439  
 setRestoreProducers  
 activemq::state::ConnectionStateTracker, 1439  
 setRestoreSessions  
 activemq::state::ConnectionStateTracker, 1439  
 setRestoreTransaction  
 activemq::state::ConnectionStateTracker, 1439  
 setResult  
 activemq::commands::IntegerResponse, 2162  
 setResume  
 activemq::commands::ConnectionControl, 1305  
 setRetroactive  
 activemq::commands::ConsumerInfo, 1507  
 setReuseAddress  
 decaf::net::ServerSocket, 3455  
 decaf::net::Socket, 3622  
 setScheduledTime  
 decaf::util::TimerTask, 3916  
 setScheme



- activemq::util::CompositeData, 1254
- decaf::internal::net::URIType, 4070
- setSchemeSpecificPart
  - decaf::internal::net::URIType, 4070
- setSeed
  - decaf::security::SecureRandom, 3428, 3429
  - decaf::util::Random, 3251
- setSelector
  - activemq::commands::ConsumerInfo, 1507
  - activemq::commands::SubscriptionInfo, 3785
- setSendBufferSize
  - decaf::net::Socket, 3622
- setSendTimeout
  - activemq::core::ActiveMQConnection, 279
  - activemq::core::ActiveMQConnectionFactory, 292
  - activemq::core::ActiveMQProducer, 474
- setServerAuthority
  - decaf::internal::net::URIType, 4070
- setServiceName
  - activemq::commands::DiscoveryEvent, 1814
- setSessionAcknowledgeMode
  - activemq::cmsutil::CmsAccessor, 1187
- setSessionId
  - activemq::commands::ConsumerId, 1476
  - activemq::commands::ProducerId, 3160
  - activemq::commands::SessionInfo, 3507
- setShort
  - activemq::commands::ActiveMQMapMessage, 363
  - activemq::util::PrimitiveList, 3078
  - activemq::util::PrimitiveMap, 3089
  - activemq::util::PrimitiveValueNode, 3114
  - cms::MapMessage, 2562
- setShortProperty
  - activemq::commands::ActiveMQMessageTemplate, 437
  - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2823
  - cms::Message, 2641
- setSize
  - activemq::commands::ProducerAck, 3127
- setSizePrefixDisabled
  - activemq::commands::WireFormatInfo, 4102
- activemq::wireformat::openwire::OpenWireFormat, 2981
- setSlaveBroker
  - activemq::commands::BrokerInfo, 908
- setSocketImplFactory
  - decaf::net::ServerSocket, 3455
  - decaf::net::Socket, 3623
- setSoLinger
  - decaf::net::Socket, 3623
- setSoTimeout
  - decaf::net::ServerSocket, 3456
  - decaf::net::Socket, 3623
- setSource
  - decaf::internal::net::URIType, 4071
- setSourceFile
  - decaf::util::logging::LogRecord, 2491
- setSourceFunction
  - decaf::util::logging::LogRecord, 2491
- setSourceLine
  - decaf::util::logging::LogRecord, 2492
- setSSLParameters
  - decaf::net::ssl::SSLSocket, 3677
- setStackTrace
  - decaf::lang::Exception, 1893
- setStackTraceElements
  - activemq::commands::BrokerError, 872
- setStackTraceEnabled
  - activemq::commands::WireFormatInfo, 4103
- activemq::wireformat::openwire::OpenWireFormat, 2981
- setStart
  - activemq::commands::ConsumerControl, 1445
- setStartupMaxReconnectAttempts
  - activemq::transport::failover::FailoverTransport, 1941
- setStop
  - activemq::commands::ConsumerControl, 1445
- setTemplate
  - decaf::util::zip::Deflater, 1768
- setMessagePropertyInterceptor
  - activemq::commands::ActiveMQMapMessage, 363
- activemq::util::PrimitiveList, 3078
- activemq::util::PrimitiveMap, 3089
- activemq::util::PrimitiveValueNode, 3114
- cms::MapMessage, 2562
- setStringProperty

activemq::commands::ActiveMQMessageTemplate, 437	activemq::commands::DestinationInfo, 1783
activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2823	activemq::transport::failover::FailoverTransport, 1941
cms::Message, 2642	setSubscriptionName, 3317
setSubscriptionName, 3317	setTimestamp, 2611
activemq::commands::RemoveSubscriptionInfo, 3785	decaf::util::logging::LogRecord, 2492
activemq::commands::SubscriptionInfo, 3785	setTimeToLive, 1108
setSubscribedDestination, 1214	activemq::cmsutil::CachedProducer, 1108
activemq::commands::SubscriptionInfo, 3785	activemq::cmsutil::CmsTemplate, 1214
setSubscriptionName, 1507	activemq::core::ActiveMQProducer, 474
activemq::commands::ConsumerInfo, 1507	cms::MessageProducer, 2817
setSubscriptionName, 2255	setTopicPrefetch, 3066
activemq::commands::JournalTopicAck, 1305	activemq::core::PrefetchPolicy, 3066
setSuspend, 1305	setTrackMessages, 1439
activemq::commands::ConnectionControl, 1305	activemq::state::ConnectionStateTracker, 1439
setSynchronizationRegistered, 310	activemq::transport::failover::FailoverTransport, 1941
activemq::core::ActiveMQConsumer, 310	setTrackTransactionProducers, 1439
setTargetConsumerId, 2611	activemq::state::ConnectionStateTracker, 1439
activemq::commands::Message, 2611	activemq::transport::failover::FailoverTransport, 1941
setTcpNoDelay, 3624	setTrackTransactions, 1439
decaf::net::Socket, 3624	activemq::state::ConnectionStateTracker, 1439
setTcpNoDelayEnabled, 4103	setTrafficClass, 3624
activemq::commands::WireFormatInfo, 4103	decaf::net::Socket, 3624
activemq::wireformat::openwire::OpenWireFormat, 2981	setTransactionId, 2255
setText, 670	activemq::commands::JournalTopicAck, 2255
activemq::commands::ActiveMQTextMessage, 3875, 3876	activemq::commands::JournalTransaction, 2310
cms::TextMessage, 3875, 3876	activemq::commands::Message, 2611
setTextView, 2754	activemq::commands::MessageAck, 2647
activemq::commands::MessageId, 2754	activemq::commands::TransactionInfo, 3963
setThrown, 2492	setTransactionState, 3216
decaf::util::logging::LogRecord, 2492	activemq::state::ProducerState, 3216
setTightEncodingEnabled, 4103	setTransport, 761
activemq::commands::WireFormatInfo, 4103	activemq::transport::failover::BackupTransport, 761
activemq::wireformat::openwire::OpenWireFormat, 2982	activemq::transport::mock::InternalCommandListener, 2193
setTime, 1722	setTransportInterruptionProcessingComplete
decaf::util::Date, 1722	
setTimeout	

- activemq::core::ActiveMQConnection, 279
- setTransportListener
  - activemq::transport::failover::FailoverTransport, 1941
  - activemq::transport::failover::FailoverTransportParentHandlers, 1941
  - activemq::transport::IOTransport, 2219
  - activemq::transport::mock::MockTransport, 2863
  - activemq::transport::Transport, 4001
  - activemq::transport::TransportFilter, 4012
- setTreadId
  - decaf::util::logging::LogRecord, 2492
- setType
  - activemq::commands::JournalTransaction, 2310
  - activemq::commands::Message, 2611
  - activemq::commands::TransactionInfo, 3963
- setUncaughtExceptionHandler
  - decaf::lang::Thread, 3884
- setupSocketImpl
  - decaf::net::ServerSocket, 3456
- setUri
  - activemq::transport::failover::BackupTransport, 761
- setUsage
  - activemq::util::MemoryUsage, 2595
- setUseAsyncSend
  - activemq::core::ActiveMQConnection, 279
  - activemq::core::ActiveMQConnectionFactory, 292
- setUseClientMode
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2929
  - decaf::internal::net::ssl::openssl::OpenSSLSSLContext, 2952
  - decaf::net::ssl::SSLContext, 3678
- setUseCollisionAvoidance
  - activemq::core::policies::DefaultRedeliveryPolicy, 1734
  - activemq::core::RedeliveryPolicy, 3272
- setUseCompression
  - activemq::core::ActiveMQConnection, 280
  - activemq::core::ActiveMQConnectionFactory, 292
- setUseExponentialBackOff
  - activemq::core::policies::DefaultRedeliveryPolicy, 1734
- activemq::core::RedeliveryPolicy, 3272
- activemq::transport::failover::FailoverTransport, 1941
- decaf::util::logging::Logger, 2473
- setUserID
  - activemq::commands::Message, 2611
- setUserInfo
  - decaf::internal::net::URIType, 4071
- setUserName
  - activemq::commands::ConnectionInfo, 1398
- setUsername
  - activemq::core::ActiveMQConnection, 280
- activemq::core::ActiveMQConnectionFactory, 292
- setValid
  - decaf::internal::net::URIType, 4071
- setValue
  - activemq::commands::BrokerId, 877
  - activemq::commands::ConnectionId, 1368
  - activemq::commands::ConsumerId, 1476
  - activemq::commands::LocalTransactionId, 2424
  - activemq::commands::MessageId, 2754
  - activemq::commands::ProducerId, 3160
  - activemq::commands::SessionId, 3480
  - activemq::util::PrimitiveValueNode, 3114
  - decaf::util::Map::Entry, 1881
- setVersion
  - activemq::commands::WireFormatInfo, 4103
  - activemq::wireformat::openwire::OpenWireFormat, 2982
  - activemq::wireformat::stomp::StompWireFormat, 3753
  - activemq::wireformat::WireFormat, 4091
- setWantClientAuth
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2929
- decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2935
- decaf::internal::net::ssl::openssl::OpenSSLSSLContext, 2953
- decaf::net::ssl::SSLParameters, 3661
- decaf::net::ssl::SSLServerSocket, 3678
- decaf::net::ssl::SSLContext, 3678
- setWasPrepared

activemq::commands::JournalTransaction, 2310  
 activemq::commands::ShutdownInfo, 3574  
 setWindowSize ShutdownInfoMarshaller  
 activemq::commands::ProducerInfo, 3189  
 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller, 3586  
 setWireFormat  
 activemq::transport::failover::FailoverTransport, 1941  
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller, 3581  
 activemq::transport::IOTransport, 2219  
 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller, 3594  
 activemq::transport::mock::MockTransport, 2863  
 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller, 3598  
 activemq::transport::Transport, 4002  
 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller, 3590  
 activemq::transport::TransportFilter, 4012  
 activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller, 3577  
 setWriteCheckTime  
 activemq::transport::inactivity::InactivityMonitor, 2069  
 SEVERE shutdownInput  
 decaf::util::logging::Level, 2408  
 decaf::internal::net::ssl::openssl::OpenSSLSocket, 2953  
 severe  
 decaf::internal::net::tcp::TcpSocket, 3859  
 decaf::net::Socket, 3625  
 decaf::net::SocketImpl, 3643  
 Short  
 decaf::lang::Short, 3541  
 SHORT\_TYPE shutdownLibrary  
 activemq::util::PrimitiveValueNode, 3104  
 activemq::library::ActiveMQCPP, 312  
 ShortArrayBuffer shutdownNetworking  
 decaf::internal::nio::ShortArrayBuffer, 3553, 3554  
 decaf::internal::net::Network, 2877  
 shutdownOutput  
 decaf::internal::net::ssl::openssl::OpenSSLSocket, 2953  
 decaf::internal::net::tcp::TcpSocket, 3859  
 decaf::net::Socket, 3625  
 decaf::net::SocketImpl, 3643  
 shutdownRuntime  
 decaf::lang::Runtime, 3421  
 signal  
 decaf::util::concurrent::locks::Condition, 1289  
 signalAll  
 decaf::util::concurrent::locks::Condition, 1289  
 SignatureException  
 decaf::security::SignatureException, 3602, 3603  
 shutdown  
 activemq::state::ConnectionState, 1432  
 activemq::state::SessionState, 3537  
 activemq::state::TransactionState, 3991  
 activemq::threads::CompositeTaskRunner, 1258  
 activemq::threads::DedicatedTaskRunner, 1725  
 activemq::threads::TaskRunner, 3849  
 ShutdownInfo  
 SimpleFormatter  
 decaf::util::logging::SimpleFormatter, 3605  
 SimpleLogger

- decaf::util::logging::SimpleLogger, 3606
- SIZE
  - decaf::lang::Byte, 979
  - decaf::lang::Character, 1135
  - decaf::lang::Double, 1853
  - decaf::lang::Float, 1973
  - decaf::lang::Integer, 2159
  - decaf::lang::Long, 2510
  - decaf::lang::Short, 3548
- size
  - activemq::commands::ProducerAck, 3128
  - activemq::core::MessageDispatchChannel, 2688
  - activemq::wireformat::openwire::utils::HexTable, 2049
  - decaf::internal::util::TimerTaskHeap, 3919
  - decaf::io::ByteArrayOutputStream, 1048
  - decaf::io::DataOutputStream, 1630
  - decaf::util::Collection, 1226
  - decaf::util::concurrent::ConcurrentStlMap, 1279
  - decaf::util::concurrent::SynchronousQueue, 3837
  - decaf::util::Map, 2549
  - decaf::util::PriorityQueue, 3124
  - decaf::util::Properties, 3224
  - decaf::util::StlList, 3707
  - decaf::util::StlMap, 3719
  - decaf::util::StlQueue, 3728
  - decaf::util::StlSet, 3737
  - gz\_state, 2041
- skip
  - decaf::internal::net::ssl::openssl::OpenSSLSocketInputOptions, 2968
  - decaf::internal::net::tcp::TcpSocketInputStream, 3862
  - decaf::io::BlockingByteArrayInputStream, 848
  - decaf::io::BufferedInputStream, 948
  - decaf::io::ByteArrayInputStream, 1046
  - decaf::io::FilterInputStream, 1956
  - decaf::io::InputStream, 2113
  - decaf::io::PushbackInputStream, 3236
  - decaf::io::Reader, 3260
  - decaf::util::zip::CheckedInputStream, 1171
  - decaf::util::zip::InflaterInputStream, 2104
  - gz\_state, 2041
- skipBytes
  - decaf::io::DataInput, 1612
  - decaf::io::DataInputStream, 1621
  - slaveBroker
    - activemq::commands::BrokerInfo, 910
  - sleep
    - decaf::lang::Thread, 3885
    - decaf::util::concurrent::TimeUnit, 3924
  - SLEEPING
    - decaf::lang::Thread, 3880
  - slice
    - decaf::internal::nio::ByteBuffer, 1037
    - decaf::internal::nio::CharArrayBuffer, 1146
    - decaf::internal::nio::DoubleArrayBuffer, 1864
    - decaf::internal::nio::FloatArrayBuffer, 1984
    - decaf::internal::nio::IntArrayBuffer, 2129
    - decaf::internal::nio::LongArrayBuffer, 2521
    - decaf::internal::nio::ShortArrayBuffer, 3559
    - decaf::nio::ByteBuffer, 1077
    - decaf::nio::CharBuffer, 1164
    - decaf::nio::DoubleBuffer, 1876
    - decaf::nio::FloatBuffer, 1996
    - decaf::nio::IntBuffer, 2141
    - decaf::nio::LongBuffer, 2533
    - decaf::nio::ShortBuffer, 3571
  - Socket
    - decaf::net::Socket, 3611–3613
  - SOCKET\_OPTION\_BINDADDR
    - decaf::net::SocketOptions, 3647
  - SOCKET\_OPTION\_BROADCAST
    - decaf::net::SocketOptions, 3647
  - SOCKET\_OPTION\_IP\_MULTICAST\_IF
    - decaf::net::SocketOptions, 3647
  - SOCKET\_OPTION\_IP\_MULTICAST\_IF2
    - decaf::net::SocketOptions, 3648
  - SOCKET\_OPTION\_IP\_MULTICAST\_LOOP
    - decaf::net::SocketOptions, 3648
  - SOCKET\_OPTION\_IP\_TOS
    - decaf::net::SocketOptions, 3648
  - SOCKET\_OPTION\_KEEPALIVE
    - decaf::net::SocketOptions, 3648
  - SOCKET\_OPTION\_LINGER
    - decaf::net::SocketOptions, 3648
  - SOCKET\_OPTION\_OOINLINE
    - decaf::net::SocketOptions, 3649

SOCKET_OPTION_RCVBUF	src/main/activemq/commands/ActiveMQBlobMessage.h,
decaf::net::SocketOptions, 3649	4186
SOCKET_OPTION_REUSEADDR	src/main/activemq/commands/ActiveMQBytesMessage.h,
decaf::net::SocketOptions, 3649	4187
SOCKET_OPTION_SNDBUF	src/main/activemq/commands/ActiveMQDestination.h,
decaf::net::SocketOptions, 3649	4188
SOCKET_OPTION_TCP_NODELAY	src/main/activemq/commands/ActiveMQMapMessage.h,
decaf::net::SocketOptions, 3650	4188
SOCKET_OPTION_TIMEOUT	src/main/activemq/commands/ActiveMQMessage.h,
decaf::net::SocketOptions, 3650	4189
SocketException	src/main/activemq/commands/ActiveMQMessageTemplate.h,
decaf::net::SocketException, 3628	4189
SocketFactory	src/main/activemq/commands/ActiveMQObjectMessage.h,
decaf::net::SocketFactory, 3631	4190
SocketFileDescriptor	src/main/activemq/commands/ActiveMQQueue.h,
decaf::internal::net::SocketFileDescriptor,	4191
3635	src/main/activemq/commands/ActiveMQStreamMessage.h,
SocketImpl	4191
decaf::net::SocketImpl, 3638	src/main/activemq/commands/ActiveMQTempDestination.h,
SocketTimeoutException	4192
decaf::net::SocketTimeoutException,	src/main/activemq/commands/ActiveMQTempQueue.h,
3651, 3652	4193
sqrt	src/main/activemq/commands/ActiveMQTempTopic.h,
decaf::lang::Math, 2588	4193
src/main/activemq/cmsutil/CachedConsumer.h,	src/main/activemq/commands/ActiveMQTextMessage.h,
4179	4194
src/main/activemq/cmsutil/CachedProducer.h,	src/main/activemq/commands/ActiveMQTopic.h,
4179	4194
src/main/activemq/cmsutil/CmsAccessor.h,	src/main/activemq/commands/BaseCommand.h,
4180	4195
src/main/activemq/cmsutil/CmsDestinationAccessor.h,	src/main/activemq/commands/BaseDataStructure.h,
4180	4195
src/main/activemq/cmsutil/CmsTemplate.h,	src/main/activemq/commands/BooleanExpression.h,
4181	4196
src/main/activemq/cmsutil/DestinationResolver.h,	src/main/activemq/commands/BrokerError.h,
4182	4196
src/main/activemq/cmsutil/DynamicDestinationResolver.h,	src/main/activemq/commands/BrokerId.h,
4182	4197
src/main/activemq/cmsutil/MessageCreator.h,	src/main/activemq/commands/BrokerInfo.h,
4183	4197
src/main/activemq/cmsutil/PooledSession.h,	src/main/activemq/commands/Command.h,
4183	4198
src/main/activemq/cmsutil/ProducerCallback.h,	src/main/activemq/commands/ConnectionControl.h,
4184	4199
src/main/activemq/cmsutil/ResourceLifecycleManager.h,	src/main/activemq/commands/ConnectionError.h,
4184	4199
src/main/activemq/cmsutil/SessionCallback.h,	src/main/activemq/commands/ConnectionId.h,
4185	4200
src/main/activemq/cmsutil/SessionPool.h,	src/main/activemq/commands/ConnectionInfo.h,
4186	4200

src/main/activemq/commands/ConsumerControl.h, 4216  
 4201 src/main/activemq/commands/PartialCommand.h,  
 src/main/activemq/commands/ConsumerId.h, 4216  
 4201 src/main/activemq/commands/ProducerAck.h,  
 src/main/activemq/commands/ConsumerInfo.h, 4217  
 4202 src/main/activemq/commands/ProducerId.h,  
 src/main/activemq/commands/ControlCommand.h, 4217  
 4203 src/main/activemq/commands/ProducerInfo.h,  
 src/main/activemq/commands/DataArrayResponse.h, 4218  
 4203 src/main/activemq/commands/RemoveInfo.h,  
 src/main/activemq/commands/DataResponse.h, 4219  
 4204 src/main/activemq/commands/RemoveSubscriptionInfo.h,  
 src/main/activemq/commands/DataSet.h, 4219  
 4204 src/main/activemq/commands/ReplayCommand.h,  
 src/main/activemq/commands/DestinationInfo.h, 4220  
 4205 src/main/activemq/commands/Response.h,  
 src/main/activemq/commands/DiscoveryEvent.h, 4220  
 4205 src/main/activemq/commands/SessionId.h,  
 src/main/activemq/commands/ExceptionResponse.h, 4221  
 4206 src/main/activemq/commands/SessionInfo.h,  
 src/main/activemq/commands/FlushCommand.h, 4221  
 4206 src/main/activemq/commands/ShutdownInfo.h,  
 src/main/activemq/commands/IntegerResponse.h, 4222  
 4207 src/main/activemq/commands/SubscriptionInfo.h,  
 src/main/activemq/commands/JournalQueueAck.h, 4222  
 4207 src/main/activemq/commands/TransactionId.h,  
 src/main/activemq/commands/JournalTopicAck.h, 4223  
 4208 src/main/activemq/commands/TransactionInfo.h,  
 src/main/activemq/commands/JournalTrace.h, 4223  
 4209 src/main/activemq/commands/WireFormatInfo.h,  
 src/main/activemq/commands/JournalTransaction.h, 4224  
 4209 src/main/activemq/commands/XATransactionId.h,  
 src/main/activemq/commands/KeepAliveInfo.h, 4224  
 4210 src/main/activemq/core/ActiveMQAckHandler.h,  
 src/main/activemq/commands/LastPartialCommand.h, 4225  
 4210 src/main/activemq/core/ActiveMQConnection.h,  
 src/main/activemq/commands/LocalTransactionId.h, 4225  
 4211 src/main/activemq/core/ActiveMQConnectionFactory.h,  
 src/main/activemq/commands/Message.h, 4211 4226  
 src/main/activemq/commands/MessageAck.h, src/main/activemq/core/ActiveMQConnectionMetaData.h,  
 4213 4227  
 src/main/activemq/commands/MessageDispatch.h, src/main/activemq/core/ActiveMQConstants.h,  
 4213 4227  
 src/main/activemq/commands/MessageDispatchNotification.h, src/main/activemq/core/ActiveMQConsumer.h,  
 4214 4228  
 src/main/activemq/commands/MessageId.h, src/main/activemq/core/ActiveMQProducer.h,  
 4215 4229  
 src/main/activemq/commands/MessagePull.h, src/main/activemq/core/ActiveMQQueueBrowser.h,  
 4215 4230  
 src/main/activemq/commands/NetworkBridgeFilter.h, src/main/activemq/core/ActiveMQSession.h,

4230  
 src/main/activemq/core/ActiveMQSessionException.h, 4251  
 4231  
 src/main/activemq/core/ActiveMQTransactionContext.h, 4252  
 src/main/activemq/core/ActiveMQTransactionManager.h, 4252  
 4232  
 src/main/activemq/core/DispatchData.h, 4233  
 src/main/activemq/core/Dispatcher.h, 4233  
 src/main/activemq/core/MessageDispatchChannel.h, 4253  
 4234  
 src/main/activemq/core/policies/DefaultPrefetchPolicy.h, 4254  
 4234  
 src/main/activemq/core/policies/DefaultRedeliveryPolicy.h, 4254  
 src/main/activemq/core/policies/RedeliveryPolicy.h, 4255  
 4235  
 src/main/activemq/core/PrefetchPolicy.h, 4255  
 src/main/activemq/core/RedeliveryPolicy.h, 4256  
 4236  
 src/main/activemq/core/Synchronization.h, 4256  
 4236  
 src/main/activemq/exceptions/ActiveMQException.h, 4257  
 4237  
 src/main/activemq/exceptions/BrokerException.h, 4257  
 4237  
 src/main/activemq/exceptions/ExceptionDefines.h, 4258  
 4238  
 src/main/activemq/io/LoggingInputStream.h, 4259  
 4242  
 src/main/activemq/io/LoggingOutputStream.h, 4260  
 4243  
 src/main/activemq/library/ActiveMQCPP.h, 4260  
 4243  
 src/main/activemq/state/CommandVisitor.h, 4261  
 4244  
 src/main/activemq/state/CommandVisitorAdapter.h, 4261  
 4244  
 src/main/activemq/state/ConnectionState.h, 4262  
 4245  
 src/main/activemq/state/ConnectionStateTracker.h, 4263  
 4246  
 src/main/activemq/state/ConsumerState.h, 4263  
 4247  
 src/main/activemq/state/ProducerState.h, 4264  
 src/main/activemq/state/SessionState.h, 4248  
 src/main/activemq/state/Tracked.h, 4249  
 src/main/activemq/state/TransactionState.h, 4249  
 4250  
 src/main/activemq/threads/CompositeTask.h, 4250  
 4250  
 src/main/activemq/threads/CompositeTaskRunner.h, 4250  
 4250  
 src/main/activemq/threads/DedicatedTaskRunner.h, 4250



4267	src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQObjectMessageMarshaller.h
src/main/activemq/transport/tcp/TcpTransportFactory.h	4306
4268	src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQQueueMarshaller.h
src/main/activemq/transport/Transport.h	4268 4310
src/main/activemq/transport/TransportFactory.h	src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQStreamMessageMarshaller.h
4269	4315
src/main/activemq/transport/TransportFilter.h	src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQTempDestinationMarshaller.h
4270	4319
src/main/activemq/transport/TransportListener.h	src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQTempQueueMarshaller.h
4270	4324
src/main/activemq/transport/TransportRegistry.h	src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQTempTopicMarshaller.h
4271	4329
src/main/activemq/util/ActiveMQProperties.h	src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQTextMessageMarshaller.h
4271	4333
src/main/activemq/util/CMSExceptionSupport.h	src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQTopicMarshaller.h
4272	4338
src/main/activemq/util/CompositeData.h	src/main/activemq/wireformat/openwire/marshall/v1/BaseCommandMarshaller.h
src/main/activemq/util/Config.h	4274 4342
src/main/activemq/util/IdGenerator.h	4275 src/main/activemq/wireformat/openwire/marshall/v1/BrokerIdMarshaller.h,
src/main/activemq/util/LongSequenceGenerator.h	4347
4276	src/main/activemq/wireformat/openwire/marshall/v1/BrokerInfoMarshaller.h,
src/main/activemq/util/MarshallingSupport.h	4351
4276	src/main/activemq/wireformat/openwire/marshall/v1/ConnectionControlMarshaller.h
src/main/activemq/util/MemoryUsage.h	4277 4355
src/main/activemq/util/PrimitiveList.h	4277 src/main/activemq/wireformat/openwire/marshall/v1/ConnectionErrorMarshaller.h
src/main/activemq/util/PrimitiveMap.h	4278 4360
src/main/activemq/util/PrimitiveValueConverter.h	src/main/activemq/wireformat/openwire/marshall/v1/ConnectionIdMarshaller.h,
4279	4364
src/main/activemq/util/PrimitiveValueNode.h	src/main/activemq/wireformat/openwire/marshall/v1/ConnectionInfoMarshaller.h
4279	4369
src/main/activemq/util/URISupport.h	4280 src/main/activemq/wireformat/openwire/marshall/v1/ConsumerControlMarshaller.h
src/main/activemq/util/Usage.h	4280 4373
src/main/activemq/wireformat/MarshalAware.h	src/main/activemq/wireformat/openwire/marshall/v1/ConsumerIdMarshaller.h,
4281	4378
src/main/activemq/wireformat/openwire/marshall/v1/BaseDataStreamMarshaller.h	src/main/activemq/wireformat/openwire/marshall/v1/ConsumerInfoMarshaller.h,
4281	4382
src/main/activemq/wireformat/openwire/marshall/v1/Data/StreamMarshaller.h	src/main/activemq/wireformat/openwire/marshall/v1/ControlCommandMarshaller.h
4282	4387
src/main/activemq/wireformat/openwire/marshall/v1/PrimitiveTypesMarshaller.h	src/main/activemq/wireformat/openwire/marshall/v1/DataArrayResponseMarshaller.h
4282	4391
src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQBlobMessageMarshaller.h	src/main/activemq/wireformat/openwire/marshall/v1/DataResponseMarshaller.h,
4283	4396
src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQByteMessageMarshaller.h	src/main/activemq/wireformat/openwire/marshall/v1/DestinationInfoMarshaller.h
4288	4400
src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQDestinationMarshaller.h	src/main/activemq/wireformat/openwire/marshall/v1/DiscoveryEventMarshaller.h
4292	4405
src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQMapMessageMarshaller.h	src/main/activemq/wireformat/openwire/marshall/v1/ExceptionResponseMarshaller.h
4297	4409
src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQMessageMarshaller.h	src/main/activemq/wireformat/openwire/marshall/v1/FlushCommandMarshaller.h
4301	4414

src/main/activemq/wireformat/openwire/marshaller/ActiveMQResponseMarshaller.h	4418	src/main/activemq/wireformat/openwire/marshaller/ActiveMQResponseMarshaller.h	4529
src/main/activemq/wireformat/openwire/marshaller/ActiveMQShutdownInfoMarshaller.h	4423	src/main/activemq/wireformat/openwire/marshaller/ActiveMQShutdownInfoMarshaller.h	4534
src/main/activemq/wireformat/openwire/marshaller/ActiveMQSubscriptionInfoMarshaller.h	4427	src/main/activemq/wireformat/openwire/marshaller/ActiveMQSubscriptionInfoMarshaller.h	4538
src/main/activemq/wireformat/openwire/marshaller/ActiveMQTransactionIdMarshaller.h	4432	src/main/activemq/wireformat/openwire/marshaller/ActiveMQTransactionIdMarshaller.h	4543
src/main/activemq/wireformat/openwire/marshaller/ActiveMQTransactionInfoMarshaller.h	4436	src/main/activemq/wireformat/openwire/marshaller/ActiveMQTransactionInfoMarshaller.h	4547
src/main/activemq/wireformat/openwire/marshaller/ActiveMQWireFormatInfoMarshaller.h	4441	src/main/activemq/wireformat/openwire/marshaller/ActiveMQWireFormatInfoMarshaller.h	4552
src/main/activemq/wireformat/openwire/marshaller/ActiveMQXATransactionInfoMarshaller.h	4445	src/main/activemq/wireformat/openwire/marshaller/ActiveMQXATransactionInfoMarshaller.h	4556
src/main/activemq/wireformat/openwire/marshaller/ActiveMQBlobMarshaller.h	4450	src/main/activemq/wireformat/openwire/marshaller/ActiveMQBlobMarshaller.h	4284
src/main/activemq/wireformat/openwire/marshaller/ActiveMQByteMarshaller.h	4454	src/main/activemq/wireformat/openwire/marshaller/ActiveMQByteMarshaller.h	4288
src/main/activemq/wireformat/openwire/marshaller/ActiveMQDeserializationMarshaller.h	4458	src/main/activemq/wireformat/openwire/marshaller/ActiveMQDeserializationMarshaller.h	4293
src/main/activemq/wireformat/openwire/marshaller/ActiveMQMapMarshaller.h	4462	src/main/activemq/wireformat/openwire/marshaller/ActiveMQMapMarshaller.h	4297
src/main/activemq/wireformat/openwire/marshaller/ActiveMQMessageMarshaller.h	4467	src/main/activemq/wireformat/openwire/marshaller/ActiveMQMessageMarshaller.h	4302
src/main/activemq/wireformat/openwire/marshaller/ActiveMQObjectMarshaller.h	4471	src/main/activemq/wireformat/openwire/marshaller/ActiveMQObjectMarshaller.h	4306
src/main/activemq/wireformat/openwire/marshaller/ActiveMQQueueMarshaller.h	4476	src/main/activemq/wireformat/openwire/marshaller/ActiveMQQueueMarshaller.h	4311
src/main/activemq/wireformat/openwire/marshaller/ActiveMQStreamMarshaller.h	4480	src/main/activemq/wireformat/openwire/marshaller/ActiveMQStreamMarshaller.h	4315
src/main/activemq/wireformat/openwire/marshaller/ActiveMQTemplateMarshaller.h	4485	src/main/activemq/wireformat/openwire/marshaller/ActiveMQTemplateMarshaller.h	4320
src/main/activemq/wireformat/openwire/marshaller/ActiveMQTextMarshaller.h	4489	src/main/activemq/wireformat/openwire/marshaller/ActiveMQTextMarshaller.h	4325
src/main/activemq/wireformat/openwire/marshaller/ActiveMQTextMarshaller.h	4494	src/main/activemq/wireformat/openwire/marshaller/ActiveMQTextMarshaller.h	4329
src/main/activemq/wireformat/openwire/marshaller/ActiveMQTextMarshaller.h	4498	src/main/activemq/wireformat/openwire/marshaller/ActiveMQTextMarshaller.h	4334
src/main/activemq/wireformat/openwire/marshaller/ActiveMQTextMarshaller.h	4503	src/main/activemq/wireformat/openwire/marshaller/ActiveMQTextMarshaller.h	4338
src/main/activemq/wireformat/openwire/marshaller/ActiveMQTextMarshaller.h	4507	src/main/activemq/wireformat/openwire/marshaller/ActiveMQTextMarshaller.h	4343
src/main/activemq/wireformat/openwire/marshaller/ActiveMQTextMarshaller.h	4512	src/main/activemq/wireformat/openwire/marshaller/ActiveMQTextMarshaller.h	4347
src/main/activemq/wireformat/openwire/marshaller/ActiveMQTextMarshaller.h	4516	src/main/activemq/wireformat/openwire/marshaller/ActiveMQTextMarshaller.h	4352
src/main/activemq/wireformat/openwire/marshaller/ActiveMQTextMarshaller.h	4521	src/main/activemq/wireformat/openwire/marshaller/ActiveMQTextMarshaller.h	4356
src/main/activemq/wireformat/openwire/marshaller/ActiveMQTextMarshaller.h	4525	src/main/activemq/wireformat/openwire/marshaller/ActiveMQTextMarshaller.h	4361

Generated on Tue Mar 8 2011 17:10:05 for activemq-cpp-3.2.5 by Doxygen

src/main/activemq/wireformat/openwire/marshaller/ActiveMQQueueMessageMarshaller	4312	4424
src/main/activemq/wireformat/openwire/marshaller/ActiveMQStoreMessageMarshaller	4316	4429
src/main/activemq/wireformat/openwire/marshaller/ActiveMQTempDestinationMarshaller	4321	4433
src/main/activemq/wireformat/openwire/marshaller/ActiveMQTempQueueMarshaller	4326	4438
src/main/activemq/wireformat/openwire/marshaller/ActiveMQTempTopicMarshaller	4330	4442
src/main/activemq/wireformat/openwire/marshaller/ActiveMQTextMessageMarshaller	4335	4447
src/main/activemq/wireformat/openwire/marshaller/ActiveMQTopicMarshaller	4339	4451
src/main/activemq/wireformat/openwire/marshaller/BasicCompoundMarshaller	4344	4455
src/main/activemq/wireformat/openwire/marshaller/BrokerIdMarshaller	4348	4459
src/main/activemq/wireformat/openwire/marshaller/BrokerInfoMarshaller	4352	4464
src/main/activemq/wireformat/openwire/marshaller/ConnectionConfirmationMarshaller	4357	4468
src/main/activemq/wireformat/openwire/marshaller/ConnectionErrorMarshaller	4361	4473
src/main/activemq/wireformat/openwire/marshaller/ConnectionInfoMarshaller	4366	4477
src/main/activemq/wireformat/openwire/marshaller/ConnectionInfoMarshaller	4370	4482
src/main/activemq/wireformat/openwire/marshaller/ConnectionInfoMarshaller	4375	4486
src/main/activemq/wireformat/openwire/marshaller/ConnectionInfoMarshaller	4379	4491
src/main/activemq/wireformat/openwire/marshaller/ConnectionInfoMarshaller	4384	4495
src/main/activemq/wireformat/openwire/marshaller/ConnectionInfoMarshaller	4388	4500
src/main/activemq/wireformat/openwire/marshaller/DataArrayResponseMarshaller	4393	4504
src/main/activemq/wireformat/openwire/marshaller/DataResponseMarshaller	4397	4509
src/main/activemq/wireformat/openwire/marshaller/DeactivationInfoMarshaller	4402	4513
src/main/activemq/wireformat/openwire/marshaller/DistortingErrorMarshaller	4406	4518
src/main/activemq/wireformat/openwire/marshaller/ExceptionResponseMarshaller	4411	4522
src/main/activemq/wireformat/openwire/marshaller/FlushCompoundMarshaller	4415	4526
src/main/activemq/wireformat/openwire/marshaller/IntegerResponseMarshaller	4420	4531

src/main/activemq/wireformat/openwire/marshaller/v3/ShutdownInfoMarshaller.h	4371
src/main/activemq/wireformat/openwire/marshaller/v3/SubscriptionInfoMarshaller.h	4376
src/main/activemq/wireformat/openwire/marshaller/v3/ActiveMQWireFormatMarshaller.h	4380
src/main/activemq/wireformat/openwire/marshaller/v3/ActiveMQWireFormatMarshaller.h	4385
src/main/activemq/wireformat/openwire/marshaller/v3/ActiveMQWireFormatMarshaller.h	4389
src/main/activemq/wireformat/openwire/marshaller/v3/ActiveMQWireFormatMarshaller.h	4394
src/main/activemq/wireformat/openwire/marshaller/v4/ActiveMQBinaryMessageMarshaller.h	4398
src/main/activemq/wireformat/openwire/marshaller/v4/ActiveMQBinaryMessageMarshaller.h	4403
src/main/activemq/wireformat/openwire/marshaller/v4/ActiveMQDestinationMarshaller.h	4407
src/main/activemq/wireformat/openwire/marshaller/v4/ActiveMQMapMessageMarshaller.h	4412
src/main/activemq/wireformat/openwire/marshaller/v4/ActiveMQMessageMarshaller.h	4416
src/main/activemq/wireformat/openwire/marshaller/v4/ActiveMQObjectMessageMarshaller.h	4421
src/main/activemq/wireformat/openwire/marshaller/v4/ActiveMQQueueInfoMarshaller.h	4425
src/main/activemq/wireformat/openwire/marshaller/v4/ActiveMQStoreMessageMarshaller.h	4430
src/main/activemq/wireformat/openwire/marshaller/v4/ActiveMQTempDestinationMarshaller.h	4434
src/main/activemq/wireformat/openwire/marshaller/v4/ActiveMQTempQueueMarshaller.h	4439
src/main/activemq/wireformat/openwire/marshaller/v4/ActiveMQTempTopicMarshaller.h	4443
src/main/activemq/wireformat/openwire/marshaller/v4/ActiveMQTextMessageMarshaller.h	4448
src/main/activemq/wireformat/openwire/marshaller/v4/ActiveMQTopicMarshaller.h	4452
src/main/activemq/wireformat/openwire/marshaller/v4/BasicCommandMarshaller.h	4456
src/main/activemq/wireformat/openwire/marshaller/v4/BufferInfoMarshaller.h	4460
src/main/activemq/wireformat/openwire/marshaller/v4/BufferInfoMarshaller.h	4464
src/main/activemq/wireformat/openwire/marshaller/v4/ConnectionControlMarshaller.h	4469
src/main/activemq/wireformat/openwire/marshaller/v4/ConnectionErrorMarshaller.h	4473
src/main/activemq/wireformat/openwire/marshaller/v4/ConnectionWireMarshaller.h	4478

src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQStream	src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQStream
4482	4318
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTemp	src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTemp
4487	4322
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTemp	src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTemp
4491	4327
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTemp	src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTemp
4496	4332
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQText	src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQText
4500	4336
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTop	src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTop
4505	4341
src/main/activemq/wireformat/openwire/marshall/v5/BaseCommand	src/main/activemq/wireformat/openwire/marshall/v5/BaseCommand
4509	4345
src/main/activemq/wireformat/openwire/marshall/v5/BrokerIdMarsh	src/main/activemq/wireformat/openwire/marshall/v5/BrokerIdMarsh
4514	4349
src/main/activemq/wireformat/openwire/marshall/v5/BrokerInfoMa	src/main/activemq/wireformat/openwire/marshall/v5/BrokerInfoMa
4518	4354
src/main/activemq/wireformat/openwire/marshall/v5/ConnectionCo	src/main/activemq/wireformat/openwire/marshall/v5/ConnectionCo
4523	4358
src/main/activemq/wireformat/openwire/marshall/v5/ConnectionErr	src/main/activemq/wireformat/openwire/marshall/v5/ConnectionErr
4527	4363
src/main/activemq/wireformat/openwire/marshall/v5/ConnectionID	src/main/activemq/wireformat/openwire/marshall/v5/ConnectionID
4532	4367
src/main/activemq/wireformat/openwire/marshall/v5/ConnectionInf	src/main/activemq/wireformat/openwire/marshall/v5/ConnectionInf
4536	4372
src/main/activemq/wireformat/openwire/marshall/v5/ConsumerCon	src/main/activemq/wireformat/openwire/marshall/v5/ConsumerCon
4541	4376
src/main/activemq/wireformat/openwire/marshall/v5/ConsumerIdM	src/main/activemq/wireformat/openwire/marshall/v5/ConsumerIdM
4545	4381
src/main/activemq/wireformat/openwire/marshall/v5/ConsumerInfo	src/main/activemq/wireformat/openwire/marshall/v5/ConsumerInfo
4550	4385
src/main/activemq/wireformat/openwire/marshall/v5/ControlComm	src/main/activemq/wireformat/openwire/marshall/v5/ControlComm
4554	4390
src/main/activemq/wireformat/openwire/marshall/v5/DataArrayRes	src/main/activemq/wireformat/openwire/marshall/v5/DataArrayRes
4559	4394
src/main/activemq/wireformat/openwire/marshall/v5/DataResponse	src/main/activemq/wireformat/openwire/marshall/v5/DataResponse
4286	4399
src/main/activemq/wireformat/openwire/marshall/v5/DestinationInf	src/main/activemq/wireformat/openwire/marshall/v5/DestinationInf
4291	4403
src/main/activemq/wireformat/openwire/marshall/v5/DiscoveryEver	src/main/activemq/wireformat/openwire/marshall/v5/DiscoveryEver
4295	4408
src/main/activemq/wireformat/openwire/marshall/v5/ExceptionResp	src/main/activemq/wireformat/openwire/marshall/v5/ExceptionResp
4300	4412
src/main/activemq/wireformat/openwire/marshall/v5/FlushComm	src/main/activemq/wireformat/openwire/marshall/v5/FlushComm
4304	4417
src/main/activemq/wireformat/openwire/marshall/v5/IntegerRespon	src/main/activemq/wireformat/openwire/marshall/v5/IntegerRespon
4309	4421
src/main/activemq/wireformat/openwire/marshall/v5/JournalQueue	src/main/activemq/wireformat/openwire/marshall/v5/JournalQueue
4313	4426

src/main/activemq/wireformat/openwire/marshaller/v5/ActiveMQInfoMarshaller.h	4430	src/main/activemq/wireformat/openwire/marshaller/v5/SubscriptionInfoMarshaller.h	4541
src/main/activemq/wireformat/openwire/marshaller/v5/ActiveMQInfoMarshaller.h	4435	src/main/activemq/wireformat/openwire/marshaller/v5/TransactionIdMarshaller.h	4546
src/main/activemq/wireformat/openwire/marshaller/v5/ActiveMQInfoMarshaller.h	4439	src/main/activemq/wireformat/openwire/marshaller/v5/TransactionInfoMarshaller.h	4550
src/main/activemq/wireformat/openwire/marshaller/v5/ActiveMQInfoMarshaller.h	4444	src/main/activemq/wireformat/openwire/marshaller/v5/WireFormatInfoMarshaller.h	4555
src/main/activemq/wireformat/openwire/marshaller/v5/ActiveMQInfoMarshaller.h	4448	src/main/activemq/wireformat/openwire/marshaller/v5/XATransactionIdMarshaller.h	4559
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQBlobMessageMarshaller.h	4453		4287
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQBytesMessageMarshaller.h	4456		4291
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQDestinationMarshaller.h	4461		4296
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQMapMessageMarshaller.h	4465		4300
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQMessageMarshaller.h	4470		4305
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQObjectMessageMarshaller.h	4474		4309
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQQueueMarshaller.h	4479		4314
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQStreamMessageMarshaller.h	4483		4319
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTempDestinationMarshaller.h	4488		4323
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTempQueueMarshaller.h	4492		4328
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTempTopicMarshaller.h	4497		4332
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTextMessageMarshaller.h	4501		4337
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTopicMarshaller.h	4506		4341
src/main/activemq/wireformat/openwire/marshaller/v6/BaseCommandMarshaller.h	4510		4346
src/main/activemq/wireformat/openwire/marshaller/v6/BrokerIdMarshaller.h	4515		4350
src/main/activemq/wireformat/openwire/marshaller/v6/BrokerInfoMarshaller.h	4519		4355
src/main/activemq/wireformat/openwire/marshaller/v6/ConnectionControlMarshaller.h	4524		4359
src/main/activemq/wireformat/openwire/marshaller/v6/ConnectionErrorMarshaller.h	4528		4364
src/main/activemq/wireformat/openwire/marshaller/v6/ConnectionIdMarshaller.h	4532		4368
src/main/activemq/wireformat/openwire/marshaller/v6/ConnectionInfoMarshaller.h	4537		4373

src/main/activemq/wireformat/openwire/marshall/v6/ConsumerControlFormat.h,marshal/v6/NetworkBridge, 4377 4488

src/main/activemq/wireformat/openwire/marshall/v6/ConsumerIdMarshaller.h,marshal/v6/PartialCommand, 4382 4493

src/main/activemq/wireformat/openwire/marshall/v6/ConsumerInfoMarshaller.h,marshal/v6/ProducerAckM, 4386 4497

src/main/activemq/wireformat/openwire/marshall/v6/ConsumerWireFormatMarshaller.h,marshal/v6/ProducerIdMa, 4391 4502

src/main/activemq/wireformat/openwire/marshall/v6/DataArrng/ResponseMarshaller.h,marshal/v6/ProducerInfoM, 4395 4506

src/main/activemq/wireformat/openwire/marshall/v6/DataResponseMarshaller.h,marshal/v6/RemoveInfoM, 4400 4511

src/main/activemq/wireformat/openwire/marshall/v6/DestinationInfoMarshaller.h,marshal/v6/RemoveSubscr, 4404 4515

src/main/activemq/wireformat/openwire/marshall/v6/ExclusiveWireFormatMarshaller.h,marshal/v6/ReplayComma, 4409 4520

src/main/activemq/wireformat/openwire/marshall/v6/ExclusiveResponseMarshaller.h,marshal/v6/ResponseMars, 4413 4524

src/main/activemq/wireformat/openwire/marshall/v6/FlushGroupWireFormatMarshaller.h,marshal/v6/SessionIdMars, 4418 4529

src/main/activemq/wireformat/openwire/marshall/v6/IntegerResponseMarshaller.h,marshal/v6/SessionInfoMa, 4422 4533

src/main/activemq/wireformat/openwire/marshall/v6/InitialQueueAckMarshaller.h,marshal/v6/ShutdownInfo, 4427 4538

src/main/activemq/wireformat/openwire/marshall/v6/InitialTopicAckMarshaller.h,marshal/v6/SubscriptionIn, 4431 4542

src/main/activemq/wireformat/openwire/marshall/v6/InitialTopicIdMarshaller.h,marshal/v6/TransactionId, 4436 4547

src/main/activemq/wireformat/openwire/marshall/v6/InitialTopicIdMarshaller.h,marshal/v6/TransactionInf, 4440 4551

src/main/activemq/wireformat/openwire/marshall/v6/KeepAliveTimeMarshaller.h,marshal/v6/WireFormatIn, 4445 4556

src/main/activemq/wireformat/openwire/marshall/v6/LastPartialCommandMarshaller.h,marshal/v6/XATransaction, 4449 4560

src/main/activemq/wireformat/openwire/marshall/v6/LastTranqswriterIdMarshaller.h,OpenWireFormat.h, 4454 4561

src/main/activemq/wireformat/openwire/marshall/v6/MarshallFactory.h,openwire/OpenWireFormatFactory.h, 4457 4562

src/main/activemq/wireformat/openwire/marshall/v6/MessageAckMarshaller.h,openwire/OpenWireFormatNegotiat, 4461 4562

src/main/activemq/wireformat/openwire/marshall/v6/MessageDispatchFormatMarshaller.h,OpenWireResponseBuilde, 4466 4563

src/main/activemq/wireformat/openwire/marshall/v6/MessageDispatchNotificationMarshaller.h,BooleanStream.h, 4470 4564

src/main/activemq/wireformat/openwire/marshall/v6/MessageIdMarshaller.h,openwire/utis/HexTable.h, 4475 4564

src/main/activemq/wireformat/openwire/marshall/v6/MessageMarshaller.h,openwire/utis/MessagePropertyInte, 4479 4565

src/main/activemq/wireformat/openwire/marshall/v6/MessagePropertyMarshaller.h,StompCommandConstants.h, 4484 4565



src/main/activemq/wireformat/stomp/StompFrame.h, 4587  
 4566 src/main/cms/Startable.h, 4587  
 src/main/activemq/wireformat/stomp/StompHelper.h, 4587  
 4567 src/main/cms/StreamMessage.h, 4588  
 src/main/activemq/wireformat/stomp/StompWireFormat.h, 4588  
 4567 src/main/cms/TemporaryQueue.h, 4588  
 src/main/activemq/wireformat/stomp/StompWireFormat.h, 4589  
 4568 src/main/cms/TemporaryTopic.h, 4589  
 src/main/activemq/wireformat/WireFormat.h, 4590  
 4569 src/main/decaf/internal/AprPool.h, 4591  
 src/main/activemq/wireformat/WireFormatFactory.h, 4591  
 4569 src/main/decaf/internal/DecafRuntime.h, 4591  
 src/main/activemq/wireformat/WireFormatNegotiator.h, 4592  
 4570 src/main/decaf/internal/io/StandardErrorOutputStream.h,  
 src/main/activemq/wireformat/WireFormatRegistry.h, 4592  
 4570 src/main/decaf/internal/io/StandardInputStream.h,  
 src/main/cms/BytesMessage.h, 4593  
 4593 src/main/decaf/internal/io/StandardOutputStream.h,  
 src/main/cms/Closeable.h, 4572 src/main/decaf/internal/net/DefaultServerSocketFactory.h,  
 src/main/cms/CMSException.h, 4573 4593  
 src/main/cms/CMSProperties.h, 4573 src/main/decaf/internal/net/DefaultSocketFactory.h,  
 src/main/cms/CMSSecurityException.h, 4574 4594  
 src/main/cms/Config.h, 4275 src/main/decaf/internal/net/Network.h, 4594  
 src/main/cms/Connection.h, 4574 src/main/decaf/internal/net/SocketFileDescriptor.h,  
 src/main/cms/ConnectionFactory.h, 4575 4595  
 src/main/cms/ConnectionMetaData.h, 4575 src/main/decaf/internal/net/ssl/DefaultSSLContext.h,  
 src/main/cms/DeliveryMode.h, 4576 4595  
 src/main/cms/Destination.h, 4576 src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h,  
 src/main/cms/ExceptionListener.h, 4576 4596  
 src/main/cms/IllegalStateException.h, 4577 src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h,  
 src/main/cms/InvalidClientIdException.h, 4578 4597  
 src/main/cms/InvalidDestinationException.h, 4578 src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h,  
 4578 4597  
 src/main/cms/InvalidSelectorException.h, 4579 src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h,  
 src/main/cms/MapMessage.h, 4579 4598  
 src/main/cms/Message.h, 4212 src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h,  
 src/main/cms/MessageConsumer.h, 4580 4598  
 src/main/cms/MessageEnumeration.h, 4580 src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h,  
 src/main/cms/MessageEOFException.h, 4581 4599  
 src/main/cms/MessageFormatException.h, 4581 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h,  
 4581 4600  
 src/main/cms/MessageListener.h, 4582 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h,  
 src/main/cms/MessageNotReadableException.h, 4600  
 4582 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h,  
 src/main/cms/MessageNotWriteableException.h, 4601  
 4583 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h,  
 src/main/cms/MessageProducer.h, 4583 4601  
 src/main/cms/ObjectMessage.h, 4584 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h,  
 src/main/cms/Queue.h, 4584 4602  
 src/main/cms/QueueBrowser.h, 4585 src/main/decaf/internal/net/tcp/TcpSocket.h,  
 src/main/cms/Session.h, 4586 4603

src/main/decaf/internal/net/tcp/TcpSocketInputStream.h,	src/main/decaf/internal/util/GenericResource.h,
4603	4618
src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h,	src/main/decaf/internal/util/HexStringParser.h,
4604	4618
src/main/decaf/internal/net/URIEncoderDecoder.h,	src/main/decaf/internal/util/Resource.h,
4604	4619
src/main/decaf/internal/net/URIHelper.h,	src/main/decaf/internal/util/ResourceLifecycleManager.h,
4605	4185
src/main/decaf/internal/net/URIType.h,	src/main/decaf/internal/util/TimerTaskHeap.h,
4605	4619
src/main/decaf/internal/nio/BufferFactory.h,	
4606	src/main/decaf/internal/util/zip/crc32.h,
	4620
src/main/decaf/internal/nio/ByteArrayBuffer.h,	src/main/decaf/internal/util/zip/deflate.h,
4607	4620
	src/main/decaf/internal/util/zip/gzguts.h,
src/main/decaf/internal/nio/CharArrayBuffer.h,	4623
4607	src/main/decaf/internal/util/zip/inffast.h,
	4625
src/main/decaf/internal/nio/DoubleArrayBuffer.h,	src/main/decaf/internal/util/zip/inffixed.h,
4608	4626
	src/main/decaf/internal/util/zip/inflate.h,
src/main/decaf/internal/nio/FloatArrayBuffer.h,	4626
4609	src/main/decaf/internal/util/zip/inftrees.h,
	4627
src/main/decaf/internal/nio/IntArrayBuffer.h,	src/main/decaf/internal/util/zip/trees.h,
4609	4628
	src/main/decaf/internal/util/zip/zconf.h,
src/main/decaf/internal/nio/LongArrayBuffer.h,	4630
4610	src/main/decaf/internal/util/zip/zlib.h,
	4632
src/main/decaf/internal/nio/ShortArrayBuffer.h,	src/main/decaf/internal/util/zip/zutil.h,
4610	4636
	src/main/decaf/io/BlockingByteArrayInputStream.h,
src/main/decaf/internal/security/unix/SecureRandomImpl.h,	4639
4611	src/main/decaf/io/BufferedInputStream.h,
	4640
src/main/decaf/internal/security/windows/SecureRandomImpl.h,	src/main/decaf/io/BufferedOutputStream.h,
4612	4640
	src/main/decaf/io/ByteArrayInputStream.h,
src/main/decaf/internal/util/ByteArrayAdapter.h,	4641
4612	src/main/decaf/io/ByteArrayOutputStream.h,
	4641
src/main/decaf/internal/util/concurrent/ConditionImpl.h,	src/main/decaf/io/Closeable.h,
4613	4572
	src/main/decaf/io/DataInput.h,
src/main/decaf/internal/util/concurrent/MutexImpl.h,	4642
4613	src/main/decaf/io/DataInputStream.h,
	4642
src/main/decaf/internal/util/concurrent/SynchronizableImpl.h,	src/main/decaf/io/DataOutput.h,
4614	4643
	src/main/decaf/io/DataOutputStream.h,
src/main/decaf/internal/util/concurrent/Transferer.h,	4644
4614	src/main/decaf/io/EOFException.h,
	4644
src/main/decaf/internal/util/concurrent/TransferQueue.h,	src/main/decaf/io/FileDescriptor.h,
4615	4645
	src/main/decaf/io/FilterInputStream.h,
src/main/decaf/internal/util/concurrent/TransferStack.h,	4645
4616	src/main/decaf/io/FilterOutputStream.h,
	4646
src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h,	src/main/decaf/io/Flushable.h,
4616	4646
	src/main/decaf/io/InputStream.h,
src/main/decaf/internal/util/concurrent/unix/MutexHandle.h,	4647
4617	src/main/decaf/io/InputStreamReader.h,
	4647
src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h,	src/main/decaf/io/InterruptedIOException.h,
4617	4648
	src/main/decaf/io/IOException.h,
src/main/decaf/internal/util/concurrent/windows/MutexHandle.h,	4648
4617	src/main/decaf/io/OutputStream.h,
	4649
src/main/decaf/internal/util/concurrent/windows/Transferer.h,	src/main/decaf/io/OutputStreamWriter.h,
4617	4649
	src/main/decaf/io/PushbackInputStream.h,
src/main/decaf/internal/util/concurrent/windows/MutexHandle.h,	4650
4617	src/main/decaf/io/Reader.h,
	4650

Generated on Tue Mar 8 2011 17:10:05 for activemq-cpp-3.2.5 by Doxygen

src/main/decaf/net/URISyntaxException.h, 4686  
 src/main/decaf/net/URL.h, 4686  
 src/main/decaf/net/URLDecoder.h, 4687  
 src/main/decaf/net/URLEncoder.h, 4687  
 src/main/decaf/nio/Buffer.h, 4687  
 src/main/decaf/nio/BufferOverflowException.h, 4688  
 src/main/decaf/nio/BufferUnderflowException.h, 4688  
 src/main/decaf/nio/ByteBuffer.h, 4689  
 src/main/decaf/nio/CharBuffer.h, 4689  
 src/main/decaf/nio/DoubleBuffer.h, 4690  
 src/main/decaf/nio/FloatBuffer.h, 4691  
 src/main/decaf/nio/IntBuffer.h, 4691  
 src/main/decaf/nio/InvalidMarkException.h, 4692  
 src/main/decaf/nio/LongBuffer.h, 4692  
 src/main/decaf/nio/ReadOnlyBufferException.h, 4693  
 src/main/decaf/nio/ShortBuffer.h, 4693  
 src/main/decaf/security/auth/x500/X500Principal.h, 4694  
 src/main/decaf/security/cert/Certificate.h, 4694  
 src/main/decaf/security/cert/CertificateEncodingException.h, 4695  
 src/main/decaf/security/cert/CertificateException.h, 4696  
 src/main/decaf/security/cert/CertificateExpiredException.h, 4696  
 src/main/decaf/security/cert/CertificateNotYetValidException.h, 4697  
 src/main/decaf/security/cert/CertificateParsingException.h, 4697  
 src/main/decaf/security/cert/X509Certificate.h, 4698  
 src/main/decaf/security/GeneralSecurityException.h, 4698  
 src/main/decaf/security/InvalidKeyException.h, 4698  
 src/main/decaf/security/Key.h, 4699  
 src/main/decaf/security/KeyException.h, 4699  
 src/main/decaf/security/KeyManagementException.h, 4700  
 src/main/decaf/security/NoSuchAlgorithmException.h, 4700  
 src/main/decaf/security/NoSuchProviderException.h, 4701  
 src/main/decaf/security/Principal.h, 4701  
 src/main/decaf/security/PublicKey.h, 4701  
 src/main/decaf/security/SecureRandom.h, 4702  
 src/main/decaf/security/SecureRandomSpi.h, 4702  
 src/main/decaf/security/SignatureException.h, 4703  
 src/main/decaf/util/AbstractCollection.h, 4703  
 src/main/decaf/util/AbstractList.h, 4704  
 src/main/decaf/util/AbstractMap.h, 4705  
 src/main/decaf/util/AbstractQueue.h, 4705  
 src/main/decaf/util/AbstractSequentialList.h, 4706  
 src/main/decaf/util/AbstractSet.h, 4707  
 src/main/decaf/util/Collection.h, 4707  
 src/main/decaf/util/Comparator.h, 4708  
 src/main/decaf/util/comparators/Less.h, 4708  
 src/main/decaf/util/concurrent/atomic/AtomicBoolean.h, 4709  
 src/main/decaf/util/concurrent/atomic/AtomicInteger.h, 4709  
 src/main/decaf/util/concurrent/atomic/AtomicReferenceCounter.h, 4710  
 src/main/decaf/util/concurrent/atomic/AtomicReference.h, 4710  
 src/main/decaf/util/concurrent/BlockingQueue.h, 4711  
 src/main/decaf/util/concurrent/BrokenBarrierException.h, 4712  
 src/main/decaf/util/concurrent/Callable.h, 4712  
 src/main/decaf/util/concurrent/CancellationException.h, 4712  
 src/main/decaf/util/concurrent/Concurrent.h, 4713  
 src/main/decaf/util/concurrent/ConcurrentMap.h, 4714  
 src/main/decaf/util/concurrent/ConcurrentStlMap.h, 4715  
 src/main/decaf/util/concurrent/CountDownLatch.h, 4715  
 src/main/decaf/util/concurrent/Delayed.h, 4716  
 src/main/decaf/util/concurrent/ExecutionException.h, 4716  
 src/main/decaf/util/concurrent/Executor.h, 4717  
 src/main/decaf/util/concurrent/ExecutorService.h, 4717  
 src/main/decaf/util/concurrent/Future.h, 4718  
 src/main/decaf/util/concurrent/Lock.h, 4718  
 src/main/decaf/util/concurrent/locks/Condition.h, 4719

---

src/main/decaf/util/concurrent/locks/Lock.h,src/main/decaf/util/logging/LoggerDefines.h,  
4719 4735

src/main/decaf/util/concurrent/locks/LockSupport.h,src/main/decaf/util/logging/LoggerHierarchy.h,  
4720 4737

src/main/decaf/util/concurrent/locks/ReadWriteLock.h,src/main/decaf/util/logging/LogManager.h,  
4721 4737

src/main/decaf/util/concurrent/locks/ReentrantLock.h,src/main/decaf/util/logging/LogRecord.h, 4738

4721 src/main/decaf/util/logging/LogWriter.h, 4739

src/main/decaf/util/concurrent/Mutex.h, 4723src/main/decaf/util/logging/MarkBlockLogger.h,  
src/main/decaf/util/concurrent/PooledThread.h, 4739

4722 src/main/decaf/util/logging/PropertiesChangeListener.h,  
src/main/decaf/util/concurrent/PooledThreadListener.h, 4740

4723 src/main/decaf/util/logging/SimpleFormatter.h,  
src/main/decaf/util/concurrent/RejectedExecutionException.h, 4740

4723 src/main/decaf/util/logging/SimpleLogger.h,  
src/main/decaf/util/concurrent/RejectedExecutionHandler.h, 4741

4724 src/main/decaf/util/logging/StreamHandler.h,  
src/main/decaf/util/concurrent/Semaphore.h, 4741

4724 src/main/decaf/util/logging/XMLFormatter.h,  
src/main/decaf/util/concurrent/Synchronizable.h, 4742

4725 src/main/decaf/util/Map.h, 4742

src/main/decaf/util/concurrent/SynchronousQueue.h,src/main/decaf/util/PriorityQueue.h, 4743

4725 src/main/decaf/util/Properties.h, 4743

src/main/decaf/util/concurrent/TaskListener.h,src/main/decaf/util/Queue.h, 4585

4726 src/main/decaf/util/Random.h, 4744

src/main/decaf/util/concurrent/ThreadFactory.h,src/main/decaf/util/Set.h, 4745

4726 src/main/decaf/util/StlList.h, 4745

src/main/decaf/util/concurrent/ThreadPool.h,src/main/decaf/util/StlMap.h, 4746

4727 src/main/decaf/util/StlQueue.h, 4747

src/main/decaf/util/concurrent/TimeoutException.h,src/main/decaf/util/StlSet.h, 4747

4728 src/main/decaf/util/StringTokenizer.h, 4748

src/main/decaf/util/concurrent/TimeUnit.h,src/main/decaf/util/Timer.h, 4748

4728 src/main/decaf/util/TimerTask.h, 4749

src/main/decaf/util/Config.h, 4275 src/main/decaf/util/UUID.h, 4750

src/main/decaf/util/Date.h, 4729 src/main/decaf/util/zip/Adler32.h, 4750

src/main/decaf/util/Iterator.h, 4729 src/main/decaf/util/zip/CheckedInputStream.h,  
4751

src/main/decaf/util/List.h, 4730 src/main/decaf/util/zip/CheckedOutputStream.h,  
src/main/decaf/util/ListIterator.h, 4730 4751

src/main/decaf/util/logging/ConsoleHandler.h,src/main/decaf/util/zip/Checksum.h, 4752

4731 src/main/decaf/util/zip/CRC32.h, 4752

src/main/decaf/util/logging/EventManager.h,src/main/decaf/util/zip/DataFormatException.h,  
4731 4753

src/main/decaf/util/logging/Filter.h, 4732 src/main/decaf/util/zip/Deflater.h, 4753

src/main/decaf/util/logging/Formatter.h, 4732src/main/decaf/util/zip/DeflaterOutputStream.h,  
src/main/decaf/util/logging/Handler.h, 4733 4754

src/main/decaf/util/logging/Level.h, 4734 src/main/decaf/util/zip/Inflater.h, 4754

src/main/decaf/util/logging/Logger.h, 4734 src/main/decaf/util/zip/InflaterInputStream.h,  
src/main/decaf/util/logging/LoggerCommon.h, 4755

4735 src/main/decaf/util/zip/ZipException.h, 4756

---

SSLContext  
     decaf::net::ssl::SSLContext, 3654  
 SSLParameters  
     decaf::net::ssl::SSLParameters, 3659, 3660  
 SSLServerSocket  
     decaf::net::ssl::SSLServerSocket, 3663, 3664  
 SSLServerSocketFactory  
     decaf::net::ssl::SSLServerSocketFactory, 3669  
 SSLSocket  
     decaf::net::ssl::SSLSocket, 3672, 3673  
 SSLSocketFactory  
     decaf::net::ssl::SSLSocketFactory, 3680  
 SslTransport  
     activemq::transport::tcp::SslTransport, 3683  
 stackTrace  
     decaf::lang::Exception, 1893  
 StandardErrorOutputStream  
     decaf::internal::io::StandardErrorOutputStream, 3687  
 StandardInputStream  
     decaf::internal::io::StandardInputStream, 3689  
 StandardOutputStream  
     decaf::internal::io::StandardOutputStream, 3690  
 start  
     activemq::commands::ConsumerControl, 1446  
     activemq::core::ActiveMQConnection, 280  
     activemq::core::ActiveMQConsumer, 310  
     activemq::core::ActiveMQSession, 531  
     activemq::core::ActiveMQSessionExecutor, 535  
     activemq::core::MessageDispatchChannel, 2688  
     activemq::transport::correlator::ResponseCorrelator, 3388  
     activemq::transport::failover::FailoverTransport, 1942  
     activemq::transport::IOTransport, 2220  
     activemq::transport::mock::MockTransport, 2863  
     activemq::transport::Transport, 4002  
     activemq::transport::TransportFilter, 4052  
     activemq::wireformat::openwire::OpenWireFormatNegotiator, 2989  
     cms::Startable, 3692  
     decaf::lang::Thread, 3885  
     gz\_state, 2041  
     startHandshake  
     decaf::internal::net::ssl::openssl::OpenSSLSocket, 2954  
     decaf::net::ssl::SSLSocket, 3678  
     stat\_desc  
     tree\_desc\_s, 4018  
     State  
     decaf::lang::Thread, 3880  
     state  
     z\_stream\_s, 4175  
     static\_dtrees  
     trees.h, 4630  
     static\_len  
     internal\_state, 2191  
     static\_ltree  
     trees.h, 4630  
     static\_tree\_desc  
     deflate.h, 4623  
     STATIC\_TREES  
     zutil.h, 4639  
     staticCast  
     decaf::lang::Pointer, 3039  
     StaticInitializer  
     activemq::core::ActiveMQConstants::StaticInitializer, 3693  
     status  
     internal\_state, 2191  
     std, 157  
     std::binary\_function, 842  
     std::less< decaf::lang::ArrayPointer< T > >, 2401  
     operator(), 2402  
     std::less< decaf::lang::Pointer< T > >, 2402  
     operator(), 2403  
     StlList  
     decaf::util::StlList, 3699, 3700  
     StlMap  
     decaf::util::StlMap, 3713  
     StlQueue  
     decaf::util::StlQueue, 3725  
     StlSet  
     decaf::util::StlSet, 3734  
     StlVector  
     decaf::util::StlVector, 3734

- activemq::wireformat::stomp::StompFrame, 3742
- activemq::util::PrimitiveValueNode::PrimitiveValue, 3098
- StompHelper
  - activemq::wireformat::stomp::StompHelper, 3747
- StompWireFormat
  - activemq::wireformat::stomp::StompWireFormat, 3752
  - internal\_state, 2191
- StompWireFormatFactory
  - subscriptionName, 3318
- activemq::wireformat::stomp::StompWireFormatFactory::commands::RemoveSubscriptionInfo, 3755
- activemq::commands::SubscriptionInfo, 3786
- stop
  - activemq::commands::ConsumerControl, 1446
  - SUBSCRIBE, 3786
- activemq::core::ActiveMQConnection, 280
- activemq::core::ActiveMQConsumer, 310
- activemq::core::ActiveMQSession, 531
- activemq::core::ActiveMQSessionExecutor, 535
- activemq::core::MessageDispatchChannel, 2688
- SubscriptionInfoMarshaller, 3783
- activemq::transport::failover::FailoverTransport, 1942
- activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller, 3792
- activemq::transport::IOTransport, 2220
- activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller, 3808
- activemq::transport::mock::MockTransport, 2864
- activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller, 3788
- activemq::transport::Transport, 4002
- activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller, 3800
- activemq::transport::TransportFilter, 4012
- activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller, 3796
- cms::Stoppable, 3756
- decaf::util::concurrent::PooledThread, 3058
- activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller, 3804
- store
  - decaf::util::Properties, 3224, 3225
- STORED
  - inflat.h, 4627
- STORED\_BLOCK
  - zutil.h, 4639
- strategy
  - gz\_state, 2041
  - internal\_state, 2191
- StreamHandler
  - decaf::util::logging::StreamHandler, 3758
- String
  - decaf::lang::String, 3777
- STRING\_TYPE
  - activemq::util::PrimitiveValueNode, 3104
  - supportsUrgentData, 3643
- StringTokenizer
  - decaf::util::StringTokenizer, 3779
- suspend

activemq::commands::ConnectionControl 1306  
 swap  
 decaf::lang::ArrayPointer, 743  
 decaf::lang::Pointer, 3040  
 decaf::util::concurrent::atomic::AtomicRefCount 756  
 SYNC  
 inflate.h, 4627  
 sync  
 decaf::io::FileDescriptor, 1948  
 SynchronizableImpl  
 decaf::internal::util::concurrent::Synchronizable 3823  
 synchronized  
 Concurrent.h, 4713  
 SynchronousQueue  
 decaf::util::concurrent::SynchronousQueue, 3830  
 syncRequest  
 activemq::core::ActiveMQConnection, 280  
 activemq::core::ActiveMQSession, 531  
 System  
 decaf::lang::System, 3840  
 TABLE  
 inflate.h, 4627  
 take  
 decaf::util::concurrent::BlockingQueue, 855  
 decaf::util::concurrent::SynchronousQueue, 3837  
 takeSession  
 activemq::cmsutil::SessionPool, 3535  
 targetConsumerId  
 activemq::commands::Message, 2613  
 Task  
 decaf::util::concurrent::ThreadPool, 3891  
 TcpSocket  
 decaf::internal::net::tcp::TcpSocket, 3854  
 TcpSocketInputStream  
 decaf::internal::net::tcp::TcpSocketInputStream, 3861  
 TcpSocketOutputStream  
 decaf::internal::net::tcp::TcpSocketOutputStream, 3864  
 TcpTransport  
 activemq::transport::tcp::TcpTransport, 3866  
 TEMP\_POSTFIX  
 activemq::commands::ActiveMQDestination, 323  
 TEMP\_PREFIX  
 activemq::commands::ActiveMQDestination, 323  
 TEMP\_QUEUE\_QUALIFIED\_PREFIX  
 activemq::commands::ActiveMQDestination, 323  
 TEMP\_TOPIC\_QUALIFIED\_PREFIX  
 activemq::commands::ActiveMQDestination, 323  
 TEMPORARY\_QUEUE  
 cms::Destination, 1777  
 TEMPORARY\_TOPIC  
 cms::Destination, 1777  
 TEMPQUEUE\_PREFIX  
 activemq::wireformat::stomp::StompCommandConstants, 3740  
 TEMPTOPIC\_PREFIX  
 activemq::wireformat::stomp::StompCommandConstants, 3740  
 TERMINATED  
 decaf::lang::Thread, 3880  
 TEXT  
 activemq::wireformat::stomp::StompCommandConstants, 3740  
 text  
 activemq::commands::ActiveMQTextMessage, 671  
 gz\_header\_s, 2039  
 Thread  
 decaf::lang::Thread, 3880, 3881  
 ThreadGroup  
 decaf::lang::ThreadGroup, 3888  
 ThreadPool  
 decaf::util::concurrent::ThreadPool, 3891  
 Throwable  
 decaf::lang::Throwable, 3896  
 Throwing  
 decaf::util::logging, 156  
 throwing  
 decaf::util::logging::Logger, 2473  
 tightMarshal  
 activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 826  
 activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1690  
 activemq::wireformat::openwire::marshal::v1::ActiveMQBlob, 196



activemq::wireformat::openwire::marshal::v1::ActiveMQBrokerMessageOpenMarshaller,	241	1800
activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller,	330	1834
activemq::wireformat::openwire::marshal::v1::ActiveMQErrorMessageMarshaller,	371	1922
activemq::wireformat::openwire::marshal::v1::ActiveMQFlushCommandMarshaller,	399	2021
activemq::wireformat::openwire::marshal::v1::ActiveMQIntegerResponseMarshaller,	447	2182
activemq::wireformat::openwire::marshal::v1::ActiveMQJournalQueueAckMarshaller,	493	2250
activemq::wireformat::openwire::marshal::v1::ActiveMQJournalTopicAckMarshaller,	560	2279
activemq::wireformat::openwire::marshal::v1::ActiveMQJournalTraceMarshaller,	589	2302
activemq::wireformat::openwire::marshal::v1::ActiveMQJournalTransactionMarshaller,	618	2334
activemq::wireformat::openwire::marshal::v1::ActiveMQKeepAliveInfoMarshaller,	652	2362
activemq::wireformat::openwire::marshal::v1::ActiveMQLastPartialCommandMarshaller,	682	2398
activemq::wireformat::openwire::marshal::v1::ActiveMQLocalTransactionIdMarshaller,	712	2448
activemq::wireformat::openwire::marshal::v1::ActiveMQMessageAckMarshaller,	789	2668
activemq::wireformat::openwire::marshal::v1::ActiveMQMessageDispatchMarshaller,	889	2710
activemq::wireformat::openwire::marshal::v1::ActiveMQMessageDispatchNotificationMarshaller,	921	2740
activemq::wireformat::openwire::marshal::v1::ActiveMQMessageIdMarshaller,	1318	2778
activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller,	1351	2800
activemq::wireformat::openwire::marshal::v1::ActiveMQMessagePullMarshaller,	1383	2848
activemq::wireformat::openwire::marshal::v1::ActiveMQNetworkBridgeFilterMarshaller,	1415	2903
activemq::wireformat::openwire::marshal::v1::ActiveMQPartialCommandMarshaller,	1462	3031
activemq::wireformat::openwire::marshal::v1::ActiveMQProducerAckMarshaller,	1492	3152
activemq::wireformat::openwire::marshal::v1::ActiveMQProducerIdMarshaller,	1525	3184
activemq::wireformat::openwire::marshal::v1::ActiveMQProducerInfoMarshaller,	1555	3201
activemq::wireformat::openwire::marshal::v1::ActiveMQRemoveInfoMarshaller,	1590	3303
activemq::wireformat::openwire::marshal::v1::ActiveMQRemoveSubscriptionInfoMarshaller,	1659	3321

activemq::wireformat::openwire::marshal::v1<ReplyCorrelationIdMessage> openwire::marshal::v2::BrokerInfoMar  
 3354 934  
 activemq::wireformat::openwire::marshal::v1<ResponseMessage> openwire::marshal::v2::ConnectionCon  
 3411 1330  
 activemq::wireformat::openwire::marshal::v1<SessionIdMessage> openwire::marshal::v2::ConnectionErro  
 3503 1338  
 activemq::wireformat::openwire::marshal::v1<SessionIdMessage> openwire::marshal::v2::ConnectionIdM  
 3519 1371  
 activemq::wireformat::openwire::marshal::v1<ShutdownWireFormatMessage> openwire::marshal::v2::ConnectionInfo  
 3587 1402  
 activemq::wireformat::openwire::marshal::v1<SubscriptionIdMessage> openwire::marshal::v2::ConsumerCont  
 3793 1449  
 activemq::wireformat::openwire::marshal::v1<TransactionIdMessage> openwire::marshal::v2::ConsumerIdMa  
 3942 1479  
 activemq::wireformat::openwire::marshal::v1<TransactionIdMessage> openwire::marshal::v2::ConsumerInfoM  
 3971 1512  
 activemq::wireformat::openwire::marshal::v1<WireFormatIdMessage> openwire::marshal::v2::ControlComma  
 4123 1542  
 activemq::wireformat::openwire::marshal::v1<XStreamIdMessage> openwire::marshal::v2::DataArrayResp  
 4162 1577  
 activemq::wireformat::openwire::marshal::v2<ActiveMQBlobMessage> openwire::marshal::v2::DataResponseM  
 205 1646  
 activemq::wireformat::openwire::marshal::v2<ActiveMQBrokerMessage> openwire::marshal::v2::DestinationInfo  
 258 1787  
 activemq::wireformat::openwire::marshal::v2<ActiveMQDestinationMessage> openwire::marshal::v2::DiscoveryEven  
 342 1822  
 activemq::wireformat::openwire::marshal::v2<ActiveMQErrorMessage> openwire::marshal::v2::ExceptionResp  
 384 1904  
 activemq::wireformat::openwire::marshal::v2<ActiveMQFlushMessage> openwire::marshal::v2::FlushCommand  
 412 2008  
 activemq::wireformat::openwire::marshal::v2<ActiveMQIntegerMessage> openwire::marshal::v2::IntegerRespons  
 460 2169  
 activemq::wireformat::openwire::marshal::v2<ActiveMQQueueMessage> openwire::marshal::v2::JournalQueueA  
 506 2234  
 activemq::wireformat::openwire::marshal::v2<ActiveMQQueueMessage> openwire::marshal::v2::JournalTopicAc  
 572 2263  
 activemq::wireformat::openwire::marshal::v2<ActiveMQQueueMessage> openwire::marshal::v2::JournalTraceM  
 601 2286  
 activemq::wireformat::openwire::marshal::v2<ActiveMQQueueMessage> openwire::marshal::v2::JournalTransac  
 630 2318  
 activemq::wireformat::openwire::marshal::v2<ActiveMQQueueMessage> openwire::marshal::v2::KeepAliveInfol  
 660 2345  
 activemq::wireformat::openwire::marshal::v2<ActiveMQQueueMessage> openwire::marshal::v2::LastPartialCom  
 695 2385  
 activemq::wireformat::openwire::marshal::v2<ActiveMQQueueMessage> openwire::marshal::v2::LocalTransactio  
 725 2431  
 activemq::wireformat::openwire::marshal::v2<BasicCommandMessage> openwire::marshal::v2::MessageAckM  
 810 2655  
 activemq::wireformat::openwire::marshal::v2<BrokerIdMessage> openwire::marshal::v2::MessageDispat  
 901 2693

activemq::wireformat::openwire::marshal::v2::ActiveMQDispatchNotificationMarshaller	395
2727	
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller	443
2758	
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	489
2791	
activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller	555
2831	
activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller	585
2883	
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller	614
3013	
activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller	644
3131	
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	674
3164	
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller	704
3197	
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller	775
3290	
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	880
3329	
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	913
3358	
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	1309
3396	
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	1343
3483	
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	1375
3528	
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	1406
3583	
activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller	1453
3809	
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	1483
3946	
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	1516
3988	
activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller	1546
4115	
activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller	1581
4154	
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	1650
192	
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	1791
237	
activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller	1826
326	
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	1909
367	

activemq::wireformat::openwire::marshal::v3::ActiveMQCommandMarshaller, openwire::marshal::v3::SessionInfoMarshaller, 2013  
 activemq::wireformat::openwire::marshal::v3::ActiveMQResponseMarshaller, openwire::marshal::v3::ShutdownInfoMarshaller, 2174  
 activemq::wireformat::openwire::marshal::v3::ActiveMQQueueAckMarshaller, openwire::marshal::v3::SubscriptionInfoMarshaller, 2242  
 activemq::wireformat::openwire::marshal::v3::ActiveMQTopicAckMarshaller, openwire::marshal::v3::TransactionIdMarshaller, 2267  
 activemq::wireformat::openwire::marshal::v3::ActiveMQTraceMarshaller, openwire::marshal::v3::TransactionInfoMarshaller, 2290  
 activemq::wireformat::openwire::marshal::v3::ActiveMQTransactionMarshaller, openwire::marshal::v3::WireFormatInfoMarshaller, 2322  
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller, openwire::marshal::v3::XATransactionInfoMarshaller, 2349  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobCommandMarshaller, openwire::marshal::v4::ActiveMQBlobMarshaller, 2380  
 activemq::wireformat::openwire::marshal::v4::ActiveMQByteArrayCommandMarshaller, openwire::marshal::v4::ActiveMQByteMarshaller, 2436  
 activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationCommandMarshaller, openwire::marshal::v4::ActiveMQDestinationMarshaller, 2659  
 activemq::wireformat::openwire::marshal::v4::ActiveMQDispatchCommandMarshaller, openwire::marshal::v4::ActiveMQDispatchMarshaller, 2697  
 activemq::wireformat::openwire::marshal::v4::ActiveMQDispatchNotificationMarshaller, openwire::marshal::v4::ActiveMQDispatchNotificationMarshaller, 2731  
 activemq::wireformat::openwire::marshal::v4::ActiveMQIDMarshaller, openwire::marshal::v4::ActiveMQIDMarshaller, 2770  
 activemq::wireformat::openwire::marshal::v4::ActiveMQQueueCommandMarshaller, openwire::marshal::v4::ActiveMQQueueMarshaller, 2787  
 activemq::wireformat::openwire::marshal::v4::ActiveMQStreamCommandMarshaller, openwire::marshal::v4::ActiveMQStreamMarshaller, 2839  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueBridgeFilterMarshaller, openwire::marshal::v4::ActiveMQTempQueueBridgeFilterMarshaller, 2895  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueCommandMarshaller, openwire::marshal::v4::ActiveMQTempQueueMarshaller, 3022  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueProducerAckMarshaller, openwire::marshal::v4::ActiveMQTempQueueProducerAckMarshaller, 3139  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueProducerWithMarshaller, openwire::marshal::v4::ActiveMQTempQueueProducerWithMarshaller, 3172  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTopicCommandMarshaller, openwire::marshal::v4::ActiveMQTopicMarshaller, 3210  
 activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller, openwire::marshal::v4::BaseCommandMarshaller, 3299  
 activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller, openwire::marshal::v4::BrokerIdMarshaller, 3325  
 activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller, openwire::marshal::v4::BrokerInfoMarshaller, 3362  
 activemq::wireformat::openwire::marshal::v4::ConnectionCommandMarshaller, openwire::marshal::v4::ConnectionCommandMarshaller, 3406  
 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller, openwire::marshal::v4::ConnectionErrorMarshaller, 3499

activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller,	1379	2844	activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller,
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller,	1411	2899	activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller,
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller,	1458	3026	activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller,
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller,	1488	3135	activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller,
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller,	1521	3168	activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller,
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller,	1550	3193	activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller,
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller,	1586	3311	activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller,
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller,	1654	3342	activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller,
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller,	1795	3349	activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller,
activemq::wireformat::openwire::marshal::v4::DiscoveryInfoMarshaller,	1830	3391	activemq::wireformat::openwire::marshal::v4::ResponseMarshaller,
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller,	1917	3487	activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller,
activemq::wireformat::openwire::marshal::v4::FlushWireFormatMarshaller,	2017	3532	activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller,
activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller,	2178	3600	activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller,
activemq::wireformat::openwire::marshal::v4::JmsQueueAckMarshaller,	2246	3801	activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller,
activemq::wireformat::openwire::marshal::v4::JmsQueueTopicAckMarshaller,	2275	3954	activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller,
activemq::wireformat::openwire::marshal::v4::JmsQueueWireFormatMarshaller,	2298	3983	activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller,
activemq::wireformat::openwire::marshal::v4::JmsQueueWireFormatMarshaller,	2330	4119	activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller,
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller,	2353	4158	activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller,
activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller,	2393	209	activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller,
activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller,	2444	250	activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller,
activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller,	2664	338	activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller,
activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller,	2706	380	activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller,
activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller,	2736	408	activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller,
activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller,	2762	456	activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller,
activemq::wireformat::openwire::marshal::v4::MessageMarshaller,	2796	502	

activemq::wireformat::openwire::marshal::v5::ActiveMQStompMessageMarshaller, marshal::v5::JournalTopicAck  
 568 2259  
 activemq::wireformat::openwire::marshal::v5::ActiveMQStompDestinationMarshaller, marshal::v5::JournalTraceMessage  
 597 2306  
 activemq::wireformat::openwire::marshal::v5::ActiveMQStompQueueOpenMarshaller, marshal::v5::JournalTransaction  
 626 2326  
 activemq::wireformat::openwire::marshal::v5::ActiveMQStompTopicMarshaller, marshal::v5::KeepAliveInfo  
 656 2358  
 activemq::wireformat::openwire::marshal::v5::ActiveMQStompMessageMarshaller, marshal::v5::LastPartialCommand  
 687 2389  
 activemq::wireformat::openwire::marshal::v5::ActiveMQStompMarshaller, wire::marshal::v5::LocalTransaction  
 716 2440  
 activemq::wireformat::openwire::marshal::v5::BasicCommandMarshaller, wire::marshal::v5::MessageAcknowledge  
 796 2672  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, openwire::marshal::v5::MessageDispatch  
 893 2701  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, openwire::marshal::v5::MessageDispatch  
 926 2744  
 activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, marshal::v5::MessageIdMarshaller  
 1322 2766  
 activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, wire::marshal::v5::MessageMarshaller  
 1355 2783  
 activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, openwire::marshal::v5::MessagePullMarshaller  
 1387 2835  
 activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, wire::marshal::v5::NetworkBridge  
 1419 2891  
 activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, wire::marshal::v5::PartialCommand  
 1466 3017  
 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller, openwire::marshal::v5::ProducerAcknowledge  
 1496 3144  
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, wire::marshal::v5::ProducerIdMarshaller  
 1529 3176  
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, wire::marshal::v5::ProducerInfoMarshaller  
 1559 3205  
 activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller, marshal::v5::RemoveInfoMarshaller  
 1594 3307  
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller, openwire::marshal::v5::RemoveSubscription  
 1637 3337  
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller, wire::marshal::v5::ReplayCommand  
 1808 3371  
 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller, wire::marshal::v5::ResponseMarshaller  
 1838 3401  
 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller, marshal::v5::SessionIdMarshaller  
 1913 3495  
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller, wire::marshal::v5::SessionInfoMarshaller  
 2025 3515  
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller, wire::marshal::v5::ShutdownInfoMarshaller  
 2187 3591  
 activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller, wire::marshal::v5::SubscriptionInfoMarshaller  
 2238 3797

activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller,	1500
3938	
activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller,	1533
3967	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	1563
4107	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	1599
4171	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	1641
213	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	1804
254	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	1817
346	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	1900
388	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	2004
416	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	2165
464	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	2230
510	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	2271
577	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	2294
605	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	2314
635	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	2341
665	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	2376
691	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	2427
721	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	2651
803	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	2714
897	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	2723
930	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	2774
1326	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	2805
1360	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	2852
1391	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	2887
1423	
activemq::wireformat::openwire::marshal::v6::WireFormatMarshaller,	3008
1470	

activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller,  
 3148  
 activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller,  
 3180  
 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller,  
 3214  
 activemq::wireformat::openwire::marshal::v6::ReceiveInfoMarshaller,  
 3294  
 activemq::wireformat::openwire::marshal::v6::ReceiveSubscriptionInfoMarshaller,  
 3333  
 activemq::wireformat::openwire::marshal::v6::ReplyCommandMarshaller,  
 3366  
 activemq::wireformat::openwire::marshal::v6::ResponseMarshaller,  
 3416  
 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller,  
 3491  
 activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller,  
 3511  
 activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller,  
 3578  
 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller,  
 3805  
 activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller,  
 3958  
 activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller,  
 3979  
 activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller,  
 4111  
 activemq::wireformat::openwire::marshal::v6::XAIdMarshaller,  
 4150  
 tightMarshal2 1526  
 activemq::wireformat::openwire::marshal::BasicDataStreamMarshaller, openwire::marshal::v1::ControlCommandMarshaller,  
 827 1555  
 activemq::wireformat::openwire::marshal::DataStreamMarshaller, openwire::marshal::v1::DataArrayResponseMarshaller,  
 1698 1591  
 activemq::wireformat::openwire::marshal::v1::ActiveMQDefaultMessageMarshaller, openwire::marshal::v1::DataResponseMarshaller,  
 197 1659  
 activemq::wireformat::openwire::marshal::v1::ActiveMQFormatMessageMarshaller, openwire::marshal::v1::DestinationInfoMarshaller,  
 242 1800  
 activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, openwire::marshal::v1::DiscoveryEventMarshaller,  
 330 1834  
 activemq::wireformat::openwire::marshal::v1::ActiveMQErrorMessageMarshaller, openwire::marshal::v1::ExceptionResponseMarshaller,  
 372 1922  
 activemq::wireformat::openwire::marshal::v1::ActiveMQFlushMessageMarshaller, openwire::marshal::v1::FlushCommandMarshaller,  
 400 2022  
 activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller, openwire::marshal::v1::IntegerResponseMarshaller,  
 448 2183  
 activemq::wireformat::openwire::marshal::v1::ActiveMQQueueInfoMarshaller, openwire::marshal::v1::JournalQueueAckMarshaller,  
 494 2251  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTextFormatMessageMarshaller, openwire::marshal::v1::JournalTopicAckMarshaller,



2280	3942
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller,	activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller,
2303	3971
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller,	activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller,
2334	4124
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller,	activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller,
2363	4163
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller,
2398	205
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller,
2449	259
activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller,
2669	342
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller,
2711	385
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller,
2741	413
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller,
2778	461
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller,
2801	507
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller,
2849	573
activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller,
2904	601
activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller,
3031	631
activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller,
3153	661
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller,
3184	696
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller,
3202	726
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller,	activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller,
3304	811
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller,	activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller,
3321	901
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller,	activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller,
3354	935
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller,	activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller,
3412	1331
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller,	activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller,
3504	1339
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller,	activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller,
3520	1371
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller,	activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller,
3588	1403
activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller,	activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller,
3794	1450
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller,	activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller,

1480	3132
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller
1513	3164
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller
1543	3198
activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller	activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller
1578	3291
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionMarshaller
1646	3330
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller
1788	3359
activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller	activemq::wireformat::openwire::marshal::v2::ResponseMarshaller
1822	3397
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller
1905	3484
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller
2009	3529
activemq::wireformat::openwire::marshal::v2::ImageResponseMarshaller	activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller
2170	3583
activemq::wireformat::openwire::marshal::v2::JmsQueueInfoMarshaller	activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller
2235	3810
activemq::wireformat::openwire::marshal::v2::JmsQueueTopicAckMarshaller	activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller
2264	3946
activemq::wireformat::openwire::marshal::v2::JmsQueueTraceMarshaller	activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller
2286	3988
activemq::wireformat::openwire::marshal::v2::JmsQueueTransactionMarshaller	activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller
2318	4116
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller	activemq::wireformat::openwire::marshal::v2::XATransactionInfoMarshaller
2346	4154
activemq::wireformat::openwire::marshal::v2::LocalPrivateCommandMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMarshaller
2385	193
activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQByteMarshaller
2432	238
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller
2656	326
activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQMapMarshaller
2694	368
activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller
2728	396
activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMarshaller
2758	444
activemq::wireformat::openwire::marshal::v2::MessageMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller
2792	490
activemq::wireformat::openwire::marshal::v2::MessagePurgeMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMarshaller
2832	556
activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller
2884	585
activemq::wireformat::openwire::marshal::v2::PartialConsumerInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller
3013	614
activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller

644	2350
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller	
675	2381
activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller	
704	2436
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller	
776	2660
activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	
881	2698
activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller	
914	2732
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller	
1310	2770
activemq::wireformat::openwire::marshal::v3::MessageMarshaller	
1343	2788
activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller	
1376	2840
activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller	
1407	2896
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller	
1454	3022
activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller	
1484	3140
activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller	
1517	3172
activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller	
1547	3210
activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller	
1582	3299
activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller	
1651	3325
activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller	
1792	3363
activemq::wireformat::openwire::marshal::v3::ResponseMarshaller	
1826	3407
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller	
1909	3500
activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller	
2013	3524
activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller	
2174	3596
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	
2243	3789
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller	
2268	3950
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	
2291	3976
activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller	
2322	4128
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller	

4167	1586
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller,marshal::v4::DataResponseM	
201	1655
activemq::wireformat::openwire::marshal::v4::ActiveMQBrokerMessageMarshaller,marshal::v4::DestinationInfo	
246	1796
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMessageMarshaller,marshal::v4::DiscoveryEven	
334	1830
activemq::wireformat::openwire::marshal::v4::ActiveMQErrorMessageMarshaller,marshal::v4::ExceptionResp	
376	1918
activemq::wireformat::openwire::marshal::v4::ActiveMQFlushMessageMarshaller,marshal::v4::FlushCommand	
404	2018
activemq::wireformat::openwire::marshal::v4::ActiveMQIntegerMessageMarshaller,marshal::v4::IntegerRespons	
452	2179
activemq::wireformat::openwire::marshal::v4::ActiveMQJournalMessageMarshaller,marshal::v4::JournalQueueA	
498	2247
activemq::wireformat::openwire::marshal::v4::ActiveMQJournalMessageMarshaller,marshal::v4::JournalTopicAc	
565	2276
activemq::wireformat::openwire::marshal::v4::ActiveMQJournalStopMessageMarshaller,marshal::v4::JournalTraceM	
593	2299
activemq::wireformat::openwire::marshal::v4::ActiveMQJournalQueueMessageMarshaller,marshal::v4::JournalTransac	
623	2330
activemq::wireformat::openwire::marshal::v4::ActiveMQJournalTopicMessageMarshaller,marshal::v4::KeepAliveInfol	
648	2354
activemq::wireformat::openwire::marshal::v4::ActiveMQLastPartialMessageMarshaller,marshal::v4::LastPartialCom	
679	2394
activemq::wireformat::openwire::marshal::v4::ActiveMQLocalTransactionMessageMarshaller,marshal::v4::LocalTransactio	
709	2445
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageAcknowledgeMessageMarshaller,marshal::v4::MessageAckM	
783	2664
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageDispatchMessageMarshaller,openwire::marshal::v4::MessageDispat	
885	2706
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageDispatchMessageMarshaller,openwire::marshal::v4::MessageDispat	
918	2736
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageIdMessageMarshaller,marshal::v4::MessageIdMar	
1314	2762
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageIdMessageMarshaller,marshal::v4::MessageMarsh	
1348	2797
activemq::wireformat::openwire::marshal::v4::ActiveMQMessagePullMessageMarshaller,marshal::v4::MessagePullM	
1380	2844
activemq::wireformat::openwire::marshal::v4::ActiveMQNetworkBridgeMessageMarshaller,marshal::v4::NetworkBridge	
1411	2900
activemq::wireformat::openwire::marshal::v4::ActiveMQPartialCommandMessageMarshaller,marshal::v4::PartialComman	
1458	3027
activemq::wireformat::openwire::marshal::v4::ActiveMQProducerAcknowledgeMessageMarshaller,marshal::v4::ProducerAckM	
1488	3136
activemq::wireformat::openwire::marshal::v4::ActiveMQProducerIdMessageMarshaller,marshal::v4::ProducerIdMar	
1521	3168
activemq::wireformat::openwire::marshal::v4::ActiveMQProducerInfoMessageMarshaller,marshal::v4::ProducerInfoM	
1551	3193
activemq::wireformat::openwire::marshal::v4::ActiveMQRemoveInfoMessageMarshaller,marshal::v4::RemoveInfoMa	

3312	797
activemq::wireformat::openwire::marshal::v4::RemoteSiteInformationOpenMarshaller,	893
3342	893
activemq::wireformat::openwire::marshal::v4::ReplyCorrelationIdOpenWire,	926
3350	926
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller,	1323
3392	1323
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller,	1356
3488	1356
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller,	1388
3533	1388
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller,	1420
3600	1420
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller,	1467
3802	1467
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller,	1496
3954	1496
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller,	1530
3984	1530
activemq::wireformat::openwire::marshal::v4::WireFormatMarshaller,	1560
4120	1560
activemq::wireformat::openwire::marshal::v4::XaTransactionIdMarshaller,	1595
4159	1595
activemq::wireformat::openwire::marshal::v5::ActiveMQJobMessageMarshaller,	1638
210	1638
activemq::wireformat::openwire::marshal::v5::ActiveMQJobMessageMarshaller,	1809
251	1809
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller,	1838
338	1838
activemq::wireformat::openwire::marshal::v5::ActiveMQJmsMessageMarshaller,	1914
380	1914
activemq::wireformat::openwire::marshal::v5::ActiveMQJmsMessageMarshaller,	2026
409	2026
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller,	2187
456	2187
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller,	2239
503	2239
activemq::wireformat::openwire::marshal::v5::ActiveMQSafeMessageMarshaller,	2259
569	2259
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller,	2307
597	2307
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicQueueMarshaller,	2326
627	2326
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller,	2358
657	2358
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMessageMarshaller,	2390
687	2390
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller,	2441
717	2441
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller,	

2673  
 activemq::wireformat::openwire::marshal::v5::ActiveMQDispatchableMarshaller  
 2702  
 activemq::wireformat::openwire::marshal::v5::ActiveMQDispatchableNotificationWireMarshaller  
 2745  
 activemq::wireformat::openwire::marshal::v5::ActiveMQIdMarshaller  
 2766  
 activemq::wireformat::openwire::marshal::v5::ActiveMQInfoMarshaller  
 2783  
 activemq::wireformat::openwire::marshal::v5::ActiveMQInfoMarshaller  
 2836  
 activemq::wireformat::openwire::marshal::v5::ActiveMQInfoMarshaller  
 2892  
 activemq::wireformat::openwire::marshal::v5::ActiveMQInfoMarshaller  
 3018  
 activemq::wireformat::openwire::marshal::v5::ActiveMQInfoMarshaller  
 3144  
 activemq::wireformat::openwire::marshal::v5::ActiveMQInfoMarshaller  
 3176  
 activemq::wireformat::openwire::marshal::v5::ActiveMQInfoMarshaller  
 3206  
 activemq::wireformat::openwire::marshal::v5::ActiveMQInfoMarshaller  
 3308  
 activemq::wireformat::openwire::marshal::v5::ActiveMQInfoMarshaller  
 3338  
 activemq::wireformat::openwire::marshal::v5::ActiveMQInfoMarshaller  
 3371  
 activemq::wireformat::openwire::marshal::v5::ActiveMQInfoMarshaller  
 3402  
 activemq::wireformat::openwire::marshal::v5::ActiveMQInfoMarshaller  
 3496  
 activemq::wireformat::openwire::marshal::v5::ActiveMQInfoMarshaller  
 3516  
 activemq::wireformat::openwire::marshal::v5::ActiveMQInfoMarshaller  
 3592  
 activemq::wireformat::openwire::marshal::v5::ActiveMQInfoMarshaller  
 3798  
 activemq::wireformat::openwire::marshal::v5::ActiveMQInfoMarshaller  
 3938  
 activemq::wireformat::openwire::marshal::v5::ActiveMQInfoMarshaller  
 3967  
 activemq::wireformat::openwire::marshal::v5::ActiveMQInfoMarshaller  
 4107  
 activemq::wireformat::openwire::marshal::v5::ActiveMQInfoMarshaller  
 4171  
 activemq::wireformat::openwire::marshal::v6::ActiveMQInfoMarshaller  
 214  
 activemq::wireformat::openwire::marshal::v6::ActiveMQInfoMarshaller  
 255  
 activemq::wireformat::openwire::marshal::v6::ActiveMQInfoMarshaller

1818  
 activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller, marshal::v6::SessionIdMarshaller,  
 1901  
 3492  
 activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller, marshal::v6::SessionInfoMarshaller,  
 2005  
 3512  
 activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller, marshal::v6::ShutdownInfoMarshaller,  
 2166  
 3579  
 activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller, marshal::v6::SubscriptionInfoMarshaller,  
 2231  
 3806  
 activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller, marshal::v6::TransactionIdMarshaller,  
 2272  
 3958  
 activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller, marshal::v6::TransactionInfoMarshaller,  
 2295  
 3980  
 activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller, marshal::v6::WireFormatInfoMarshaller,  
 2314  
 4112  
 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller, marshal::v6::XATransactionIdMarshaller,  
 2341  
 4150  
 activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller, tightMarshalBrokerError1  
 2377  
 activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller, tightMarshalBrokerError2  
 2428  
 activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller, tightMarshalCachedObject1  
 2652  
 828  
 activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller, tightMarshalCachedObject2  
 2715  
 828  
 activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller, tightMarshalLong1  
 2724  
 828  
 activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller, tightMarshalLong2  
 2774  
 829  
 activemq::wireformat::openwire::marshal::v6::MessageMarshaller, tightMarshalNestedObject1  
 2806  
 829  
 activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller, tightMarshalNestedObject2  
 2853  
 830  
 activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller, tightMarshalObjectArray1  
 2888  
 831  
 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, tightMarshalResponseMarshaller,  
 3009  
 830  
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller, tightMarshalResponseMarshaller,  
 3149  
 2982  
 activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, tightMarshalResponseMarshaller,  
 3180  
 2982  
 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller, tightMarshalResponseMarshaller,  
 3215  
 830  
 activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller, tightMarshalResponseMarshaller,  
 3295  
 2982  
 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller, tightMarshalResponseMarshaller,  
 3334  
 831  
 activemq::wireformat::openwire::marshal::v6::ReplyCommandMarshaller, tightMarshalResponseMarshaller,  
 3367  
 831  
 activemq::wireformat::openwire::marshal::v6::ReplyResponseMarshaller, tightMarshalResponseMarshaller,

activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 1488  
 831  
 activemq::wireformat::openwire::marshal::v1::ConnectionInfo, 1416  
 tightMarshalString1, 1416  
 activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 1463  
 831  
 tightMarshalString2, 1463  
 activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller, 1468  
 activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 1488  
 832  
 activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller, 1488  
 tightUnmarshal, 1526  
 activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 1556  
 832  
 activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller, 1556  
 activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1591  
 1705  
 activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller, 1591  
 197  
 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1660  
 197  
 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller, 1801  
 242  
 activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1835  
 331  
 activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller, 1923  
 372  
 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller, 2022  
 400  
 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller, 2183  
 448  
 activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller, 2251  
 494  
 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller, 2280  
 561  
 activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller, 2303  
 590  
 activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller, 2335  
 619  
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller, 2363  
 653  
 activemq::wireformat::openwire::marshal::v1::LastPartialCommitMarshaller, 2399  
 683  
 activemq::wireformat::openwire::marshal::v1::LocalTransactionMarshaller, 2449  
 713  
 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller, 2669  
 791  
 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2711  
 890  
 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2741  
 922  
 activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 2779  
 1319  
 activemq::wireformat::openwire::marshal::v1::MessageMarshaller, 2802  
 1352  
 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller, 2802



2849	573
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller
2904	602
activemq::wireformat::openwire::marshal::v1::PartialConsumerIdMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller
3032	631
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller
3153	661
activemq::wireformat::openwire::marshal::v1::ProducerWithMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller
3185	696
activemq::wireformat::openwire::marshal::v1::ProducerWithMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller
3202	726
activemq::wireformat::openwire::marshal::v1::RemoteWireMarshaller	activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller
3304	812
activemq::wireformat::openwire::marshal::v1::RemoteWireMarshaller	activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller
3321	902
activemq::wireformat::openwire::marshal::v1::ReplyConsumerIdMarshaller	activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller
3355	935
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller
3413	1331
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller	activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller
3504	1339
activemq::wireformat::openwire::marshal::v1::SessionWireMarshaller	activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller
3520	1372
activemq::wireformat::openwire::marshal::v1::StartupWireMarshaller	activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller
3588	1403
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller
3794	1450
activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller	activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller
3943	1480
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller
3972	1513
activemq::wireformat::openwire::marshal::v1::WireForwardMarshaller	activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller
4124	1543
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller	activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller
4163	1578
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller	activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller
206	1647
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller	activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller
259	1788
activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller	activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller
343	1823
activemq::wireformat::openwire::marshal::v2::ActiveMQErrorMarshaller	activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller
385	1905
activemq::wireformat::openwire::marshal::v2::ActiveMQErrorMessageMarshaller	activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller
413	2009
activemq::wireformat::openwire::marshal::v2::ActiveMQIntegerResponseMarshaller	activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller
461	2170
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller
507	2235
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMessageMarshaller	activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller

2264	3947
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller
2287	3989
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller
2319	4116
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::XATransactionInfoMarshaller
2346	4155
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMarshaller
2386	193
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQByteMarshaller
2432	238
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller
2656	327
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQMapMarshaller
2694	368
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller
2728	396
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMarshaller
2758	444
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller
2793	490
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMarshaller
2832	556
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller
2884	586
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller
3014	614
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller
3132	644
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQTextMarshaller
3164	675
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller
3198	705
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller
3291	777
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller
3330	881
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller
3359	914
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ConnectionConfirmationMarshaller
3398	1310
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller
3484	1344
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller
3529	1376
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller
3584	1407
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ConsumerContextMarshaller
3810	1454
activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller

1484	3140
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller,	activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller,
1517	3173
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller,	activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller,
1547	3211
activemq::wireformat::openwire::marshal::v3::DeleteWireResponseMarshaller,	activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller,
1582	3300
activemq::wireformat::openwire::marshal::v3::DeleteResponseMarshaller,	activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller,
1651	3326
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller,	activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller,
1792	3363
activemq::wireformat::openwire::marshal::v3::DiscoveryInfoMarshaller,	activemq::wireformat::openwire::marshal::v3::ResponseMarshaller,
1827	3408
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller,	activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller,
1910	3500
activemq::wireformat::openwire::marshal::v3::FlushWireInfoMarshaller,	activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller,
2014	3525
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller,	activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller,
2175	3596
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller,	activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller,
2243	3790
activemq::wireformat::openwire::marshal::v3::MessageTopicAckMarshaller,	activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller,
2268	3951
activemq::wireformat::openwire::marshal::v3::MessageTraceMarshaller,	activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller,
2291	3976
activemq::wireformat::openwire::marshal::v3::MessageTraceMarshaller,	activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller,
2323	4128
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller,	activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller,
2350	4167
activemq::wireformat::openwire::marshal::v3::LocalPartialCommandMarshaller,	activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller,
2381	201
activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller,	activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller,
2437	247
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller,	activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller,
2660	335
activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller,	activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller,
2698	376
activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller,	activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller,
2732	405
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller,	activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller,
2771	452
activemq::wireformat::openwire::marshal::v3::MessageMarshaller,	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller,
2788	499
activemq::wireformat::openwire::marshal::v3::MessagePushMarshaller,	activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller,
2840	565
activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilteredMarshaller,	activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller,
2896	594
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller,	activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller,
3023	623
activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller,	activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller,

649	2354
activemq::wireformat::openwire::marshal::v4::ActiveMQInfoMessageMarshaller;marshal::v4::LastPartialCom	
679	2394
activemq::wireformat::openwire::marshal::v4::ActiveMQInfoMessageMarshaller;marshal::v4::LocalTransactio	
709	2445
activemq::wireformat::openwire::marshal::v4::BaseQwinnfoMarshaller;marshal::v4::MessageAckM	
784	2665
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller;openwire::marshal::v4::MessageDispat	
885	2707
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller;openwire::marshal::v4::MessageDispat	
918	2737
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller;marshal::v4::MessageIdMar	
1314	2763
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller;marshal::v4::MessageMarsh	
1348	2797
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller;openwire::marshal::v4::MessagePullMa	
1380	2845
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller;openwire::marshal::v4::NetworkBridge	
1412	2900
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller;marshal::v4::PartialComman	
1459	3027
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller;openwire::marshal::v4::ProducerAckM	
1489	3136
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller;openwire::marshal::v4::ProducerIdMar	
1522	3169
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller;openwire::marshal::v4::ProducerInfoM	
1551	3194
activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller;marshal::v4::RemoveInfoMa	
1587	3312
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller;openwire::marshal::v4::RemoveSubscri	
1655	3343
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller;openwire::marshal::v4::ReplayComm	
1796	3350
activemq::wireformat::openwire::marshal::v4::DiscoveryResponseMarshaller;openwire::marshal::v4::ResponseMarsh	
1831	3393
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller;marshal::v4::SessionIdMarsh	
1918	3488
activemq::wireformat::openwire::marshal::v4::FlushQwinnfoMarshaller;openwire::marshal::v4::SessionInfoMar	
2018	3533
activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller;openwire::marshal::v4::ShutdownInfoM	
2179	3601
activemq::wireformat::openwire::marshal::v4::JmsQlQueueAckMarshaller;openwire::marshal::v4::SubscriptionInf	
2247	3802
activemq::wireformat::openwire::marshal::v4::JmsQlTopicAckMarshaller;openwire::marshal::v4::TransactionIdM	
2276	3955
activemq::wireformat::openwire::marshal::v4::JmsQlTraceMarshaller;openwire::marshal::v4::TransactionInfo	
2299	3984
activemq::wireformat::openwire::marshal::v4::JmsQlTransactionMarshaller;marshal::v4::WireFormatInfo	
2331	4120
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller;openwire::marshal::v4::XATransactionI	

4159	1595
activemq::wireformat::openwire::marshal::v5::ActiveMQBrokerMessageMarshaller,marshal::v5::DataResponseMarshaller,	
210	1638
activemq::wireformat::openwire::marshal::v5::ActiveMQBrokerMessageMarshaller,marshal::v5::DestinationInfoMarshaller,	
251	1809
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller,marshal::v5::DiscoveryEventMarshaller,	
339	1839
activemq::wireformat::openwire::marshal::v5::ActiveMQErrorMessageMarshaller,marshal::v5::ExceptionResponseMarshaller,	
381	1914
activemq::wireformat::openwire::marshal::v5::ActiveMQFlushCommandMarshaller,marshal::v5::FlushCommandMarshaller,	
409	2026
activemq::wireformat::openwire::marshal::v5::ActiveMQIntegerResponseMarshaller,marshal::v5::IntegerResponseMarshaller,	
457	2188
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalQueueAckMarshaller,marshal::v5::JournalQueueAckMarshaller,	
503	2239
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTopicAckMarshaller,marshal::v5::JournalTopicAckMarshaller,	
569	2260
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTraceMarshaller,marshal::v5::JournalTraceMarshaller,	
598	2307
activemq::wireformat::openwire::marshal::v5::ActiveMQJournalTransactionMarshaller,marshal::v5::JournalTransactionMarshaller,	
627	2327
activemq::wireformat::openwire::marshal::v5::ActiveMQKeepAliveInfoMarshaller,marshal::v5::KeepAliveInfoMarshaller,	
657	2359
activemq::wireformat::openwire::marshal::v5::ActiveMQLastPartialCommandMarshaller,marshal::v5::LastPartialCommandMarshaller,	
688	2390
activemq::wireformat::openwire::marshal::v5::ActiveMQLocalTransactionIdMarshaller,marshal::v5::LocalTransactionIdMarshaller,	
717	2441
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageAckMarshaller,marshal::v5::MessageAckMarshaller,	
798	2673
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageDispatchMarshaller,marshal::v5::MessageDispatchMarshaller,	
894	2702
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageDispatchNotificationMarshaller,marshal::v5::MessageDispatchNotificationMarshaller,	
927	2745
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageIdMarshaller,marshal::v5::MessageIdMarshaller,	
1323	2767
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller,marshal::v5::MessageMarshaller,	
1356	2784
activemq::wireformat::openwire::marshal::v5::ActiveMQMessagePullMarshaller,marshal::v5::MessagePullMarshaller,	
1388	2836
activemq::wireformat::openwire::marshal::v5::ActiveMQNetworkBridgeFilterMarshaller,marshal::v5::NetworkBridgeFilterMarshaller,	
1420	2892
activemq::wireformat::openwire::marshal::v5::ActiveMQPartialCommandMarshaller,marshal::v5::PartialCommandMarshaller,	
1467	3018
activemq::wireformat::openwire::marshal::v5::ActiveMQProducerAckMarshaller,marshal::v5::ProducerAckMarshaller,	
1497	3145
activemq::wireformat::openwire::marshal::v5::ActiveMQProducerIdMarshaller,marshal::v5::ProducerIdMarshaller,	
1530	3177
activemq::wireformat::openwire::marshal::v5::ActiveMQProducerInfoMarshaller,marshal::v5::ProducerInfoMarshaller,	
1560	3206
activemq::wireformat::openwire::marshal::v5::ActiveMQRemoveInfoMarshaller,marshal::v5::RemoveInfoMarshaller,	

3308	805
activemq::wireformat::openwire::marshal::v6::ActiveMQSubscriptionInfoMarshaller	805
3338	898
activemq::wireformat::openwire::marshal::v6::ActiveMQBrokerInfoMarshaller	898
3372	931
activemq::wireformat::openwire::marshal::v6::ActiveMQConnectionInfoMarshaller	931
3403	1327
activemq::wireformat::openwire::marshal::v6::ActiveMQConnectionErrorMarshaller	1327
3496	1361
activemq::wireformat::openwire::marshal::v6::ActiveMQConnectionIdMarshaller	1361
3516	1392
activemq::wireformat::openwire::marshal::v6::ActiveMQConnectionInfoMarshaller	1392
3592	1424
activemq::wireformat::openwire::marshal::v6::ActiveMQConsumerInfoMarshaller	1424
3798	1471
activemq::wireformat::openwire::marshal::v6::ActiveMQConsumerIdMarshaller	1471
3939	1501
activemq::wireformat::openwire::marshal::v6::ActiveMQConsumerInfoMarshaller	1501
3967	1534
activemq::wireformat::openwire::marshal::v6::ActiveMQControlCommandMarshaller	1534
4108	1564
activemq::wireformat::openwire::marshal::v6::ActiveMQDataArrayResponseMarshaller	1564
4172	1600
activemq::wireformat::openwire::marshal::v6::ActiveMQDataResponseMarshaller	1600
214	1642
activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationInfoMarshaller	1642
255	1805
activemq::wireformat::openwire::marshal::v6::ActiveMQDiscoveryEventMarshaller	1805
347	1818
activemq::wireformat::openwire::marshal::v6::ActiveMQExceptionResponseMarshaller	1818
389	1901
activemq::wireformat::openwire::marshal::v6::ActiveMQFlushCommandMarshaller	1901
417	2005
activemq::wireformat::openwire::marshal::v6::ActiveMQIntegerResponseMarshaller	2005
465	2166
activemq::wireformat::openwire::marshal::v6::ActiveMQJournalQueueAckMarshaller	2166
511	2231
activemq::wireformat::openwire::marshal::v6::ActiveMQJournalTopicAckMarshaller	2231
578	2272
activemq::wireformat::openwire::marshal::v6::ActiveMQJournalTraceMarshaller	2272
606	2295
activemq::wireformat::openwire::marshal::v6::ActiveMQJournalTransactionMarshaller	2295
636	2314
activemq::wireformat::openwire::marshal::v6::ActiveMQKeepAliveInfoMarshaller	2314
666	2342
activemq::wireformat::openwire::marshal::v6::ActiveMQLastPartialCommandMarshaller	2342
692	2377
activemq::wireformat::openwire::marshal::v6::ActiveMQLocalTransactionMarshaller	2377
722	2428
activemq::wireformat::openwire::marshal::v6::ActiveMQMessageAckMarshaller	2428

- 2652
- activemq::wireformat::openwire::marshal::tightUnmarshalDispatchNotificationMarshaller, 833
- 2715
- activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller, 834
- 2724
- tightUnmarshalConstByteArray
- activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller, 834
- 2775
- activemq::wireformat::openwire::marshal::tightUnmarshalMsgMarshaller, 834
- 2806
- activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller, 834
- 2853
- tightUnmarshalNestedObject
- activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller, 835
- 2888
- activemq::wireformat::openwire::marshal::v6::PartialWireFormatMarshaller, 835
- 3009
- activemq::wireformat::openwire::marshal::tightUnmarshalSessionMarshaller, 2983
- 3149
- activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, 2983
- 3181
- TIME
- activemq::wireformat::openwire::marshal::v6::TimeUnitMarshaller, 4026
- 3215
- time
- activemq::wireformat::openwire::marshal::v6::TimerImplMarshaller, 3904
- 3295
- TIMED\_WAITING
- activemq::wireformat::openwire::marshal::v6::RefLangSubscriptionInfoMarshaller, 3880
- 3334
- timedJoin
- activemq::wireformat::openwire::marshal::v6::ReplyCommandMarshaller, 3925
- 3367
- timedWait
- activemq::wireformat::openwire::marshal::v6::ResponseMarshaller, 3925
- 3418
- timeout
- activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller, 1784
- 3492
- activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 2828
- 3512
- TimeoutException
- activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller, 3900, 3901
- 3579
- TimeUnit
- activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller, 3904
- 3806
- decaf::util::Timer, 3904
- activemq::wireformat::openwire::marshal::v6::TimerTaskMarshaller, 3916
- 3959
- TimerImpl
- activemq::wireformat::openwire::marshal::v6::TimerTaskMarshaller, 3916
- 3980
- TimerTask
- activemq::wireformat::openwire::marshal::v6::TimerTaskMarshaller, 3916
- 4112
- TimerTaskHeap
- activemq::wireformat::openwire::marshal::v6::TimerTaskMarshaller, 3918
- 4150
- timestamp
- tightUnmarshalBrokerError
- activemq::wireformat::openwire::marshal::v6::MessageMarshaller, 2613
- 833
- decaf::util::UUID, 4086
- tightUnmarshalByteArray
- TimeUnit
- activemq::wireformat::openwire::marshal::v6::MessageMarshaller, 3922

- toArray
  - activemq::util::ActiveMQProperties, 478
  - cms::CMSProperties, 1199
  - decaf::util::AbstractCollection, 170
  - decaf::util::Collection, 1226
  - decaf::util::concurrent::SynchronousQueue, 3838
  - decaf::util::Properties, 3226
  - decaf::util::StlQueue, 3728
  - decaf::util::StringTokenizer, 3781
- toBinaryString
  - decaf::lang::Integer, 2156
  - decaf::lang::Long, 2507
- toByteArray
  - decaf::io::ByteArrayOutputStream, 1049
- toDays
  - decaf::util::concurrent::TimeUnit, 3926
- toDegrees
  - decaf::lang::Math, 2591
- toDestinationOption
  - activemq::core::ActiveMQConstants, 300
- toHexFromBytes
  - activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 836
- toHexString
  - decaf::lang::Double, 1851
  - decaf::lang::Float, 1971
  - decaf::lang::Integer, 2156
  - decaf::lang::Long, 2508
- toHours
  - decaf::util::concurrent::TimeUnit, 3926
- toMicros
  - decaf::util::concurrent::TimeUnit, 3926
- toMillis
  - decaf::util::concurrent::TimeUnit, 3927
- toMinutes
  - decaf::util::concurrent::TimeUnit, 3927
- toNanos
  - decaf::util::concurrent::TimeUnit, 3928
- toOctalString
  - decaf::lang::Integer, 2156
  - decaf::lang::Long, 2508
- TOPIC
  - cms::Destination, 1777
- TOPIC\_PREFIX
  - activemq::wireformat::stomp::StompCommandConstants, 3740
- TOPIC\_QUALIFIED\_PREFIX
  - activemq::commands::ActiveMQDestination, 323
  - toRadians
  - decaf::lang::Math, 2592
  - toSeconds
  - decaf::util::concurrent::TimeUnit, 3928
  - toStream
    - activemq::wireformat::stomp::StompFrame, 3745
  - toString
    - activemq::commands::ActiveMQBlobMessage, 189
    - activemq::commands::ActiveMQBytesMessage, 229
    - activemq::commands::ActiveMQDestination, 321
    - activemq::commands::ActiveMQMapMessage, 364
    - activemq::commands::ActiveMQMessage, 392
    - activemq::commands::ActiveMQObjectMessage, 440
    - activemq::commands::ActiveMQQueue, 488
    - activemq::commands::ActiveMQStreamMessage, 547
    - activemq::commands::ActiveMQTempDestination, 581
    - activemq::commands::ActiveMQTempQueue, 610
    - activemq::commands::ActiveMQTempTopic, 640
    - activemq::commands::ActiveMQTextMessage, 671
    - activemq::commands::ActiveMQTopic, 700
    - activemq::commands::BaseCommand, 770
    - activemq::commands::BaseDataStructure, 841
    - activemq::commands::BooleanExpression, 863
    - activemq::commands::BrokerId, 877
    - activemq::commands::BrokerInfo, 908
    - activemq::commands::Command, 1231
    - activemq::commands::ConnectionControl, 1305
    - activemq::commands::ConnectionError, 1335



- activemq::commands::ConnectionId, 1368
- activemq::commands::ConnectionInfo, 1398
- activemq::commands::ConsumerControl, 1445
- activemq::commands::ConsumerId, 1476
- activemq::commands::ConsumerInfo, 1507
- activemq::commands::ControlCommand, 1538
- activemq::commands::DataArrayResponse, 1574
- activemq::commands::DataResponse, 1634
- activemq::commands::DataStructure, 1718
- activemq::commands::DestinationInfo, 1783
- activemq::commands::DiscoveryEvent, 1814
- activemq::commands::ExceptionResponse, 1897
- activemq::commands::FlushCommand, 2001
- activemq::commands::IntegerResponse, 2162
- activemq::commands::JournalQueueAck, 2227
- activemq::commands::JournalTopicAck, 2255
- activemq::commands::JournalTrace, 2283
- activemq::commands::JournalTransaction, 2310
- activemq::commands::KeepAliveInfo, 2337
- activemq::commands::LastPartialCommand, 2373
- activemq::commands::LocalTransactionId, 2424
- activemq::commands::Message, 2611
- activemq::commands::MessageAck, 2647
- activemq::commands::MessageDispatch, 2682
- activemq::commands::MessageDispatchNotification, 2719
- activemq::commands::MessageId, 2754
- activemq::commands::MessagePull, 2827
- activemq::commands::NetworkBridgeFilter, 2880
- activemq::commands::PartialCommand, 3005
- activemq::commands::ProducerAck, 3127
- activemq::commands::ProducerId, 3160
- activemq::commands::ProducerInfo, 3189
- activemq::commands::RemoveInfo, 3287
- activemq::commands::RemoveSubscriptionInfo, 3317
- activemq::commands::ReplayCommand, 3346
- activemq::commands::Response, 3382
- activemq::commands::SessionId, 3480
- activemq::commands::SessionInfo, 3507
- activemq::commands::ShutdownInfo, 3575
- activemq::commands::SubscriptionInfo, 3785
- activemq::commands::TransactionId, 3935
- activemq::commands::TransactionInfo, 3963
- activemq::commands::WireFormatInfo, 4103
- activemq::commands::XATransactionId, 4146
- activemq::core::ActiveMQConstants, 300
- activemq::state::ConnectionState, 1432
- activemq::state::ConsumerState, 1536
- activemq::state::ProducerState, 3216
- activemq::state::SessionState, 3537
- activemq::state::TransactionState, 3991
- activemq::util::ActiveMQProperties, 479
- activemq::util::PrimitiveList, 3079
- activemq::util::PrimitiveMap, 3089
- activemq::util::PrimitiveValueNode, 3114
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 836, 837
- cms::CMSProperties, 1199
- decaf::io::ByteArrayOutputStream, 1049
- decaf::io::FilterOutputStream, 1961
- decaf::io::InputStream, 2114
- decaf::io::OutputStream, 2995
- decaf::lang::Boolean, 860
- decaf::lang::Byte, 977
- decaf::lang::Character, 1134
- decaf::lang::CharSequence, 1169
- decaf::lang::Double, 1851
- decaf::lang::Float, 1971, 1972
- decaf::lang::Integer, 2157, 2158

- decaf::lang::Long, 2508, 2509
- decaf::lang::Short, 3546, 3547
- decaf::lang::String, 3778
- decaf::lang::Thread, 3886
- decaf::net::InetAddress, 2084
- decaf::net::ServerSocket, 3456
- decaf::net::Socket, 3625
- decaf::net::SocketImpl, 3644
- decaf::net::URI, 4043
- decaf::nio::ByteBuffer, 1078
- decaf::nio::CharBuffer, 1165
- decaf::nio::DoubleBuffer, 1876
- decaf::nio::FloatBuffer, 1996
- decaf::nio::IntBuffer, 2141
- decaf::nio::LongBuffer, 2533
- decaf::nio::ShortBuffer, 3571
- decaf::security::cert::Certificate, 1114
- decaf::util::concurrent::atomic::AtomicBoolean, 748
- decaf::util::concurrent::atomic::AtomicInteger, 753
- decaf::util::concurrent::atomic::AtomicReference, 758
- decaf::util::concurrent::locks::ReentrantLock, 3278
- decaf::util::concurrent::Semaphore, 3442
- decaf::util::concurrent::TimeUnit, 3928
- decaf::util::Date, 1723
- decaf::util::logging::Level, 2407
- decaf::util::Properties, 3226
- decaf::util::UUID, 4087
- total
  - inflate\_state, 2087
- total\_in
  - z\_stream\_s, 4175
- total\_out
  - z\_stream\_s, 4175
- toURI
  - activemq::util::CompositeData, 1254
- toURIOption
  - activemq::core::ActiveMQConstants, 300
- toURL
  - decaf::net::URI, 4043
- Trace
  - zutil.h, 4639
- Tracec
  - zutil.h, 4639
- Tracecv
  - zutil.h, 4639
- Tracev
  - zutil.h, 4639
- Tracevv
  - zutil.h, 4639
- track
  - activemq::state::ConnectionStateTracker, 1439
- trackBack
  - activemq::state::ConnectionStateTracker, 1439
- Tracked
  - activemq::state::Tracked, 3932
- TRANSACTION\_STATE\_BEGIN
  - activemq::core::ActiveMQConstants, 299
- TRANSACTION\_STATE\_COMMITONEPHASE
  - activemq::core::ActiveMQConstants, 299
- TRANSACTION\_STATE\_COMMITTWOPHASE
  - activemq::core::ActiveMQConstants, 299
- TRANSACTION\_STATE\_END
  - activemq::core::ActiveMQConstants, 299
- TRANSACTION\_STATE\_FORGET
  - activemq::core::ActiveMQConstants, 299
- TRANSACTION\_STATE\_PREPARE
  - activemq::core::ActiveMQConstants, 299
- TRANSACTION\_STATE\_RECOVER
  - activemq::core::ActiveMQConstants, 299
- TRANSACTION\_STATE\_ROLLBACK
  - activemq::core::ActiveMQConstants, 299
- TransactionId
  - activemq::commands::TransactionId, 3933
- transactionId
  - activemq::commands::JournalTopicAck, 2256
  - activemq::commands::JournalTransaction, 2311
  - activemq::commands::Message, 2613
  - activemq::commands::MessageAck, 2648
  - activemq::commands::TransactionInfo, 3964
- TransactionIdMarshaller

Generated on Tue Mar 8 2011 17:10:05 for activemq-cpp-3.2.5 by Doxygen

- decaf::util::concurrent::Synchronizable, 3815
- decaf::util::StlMap, 3719
- decaf::util::StlQueue, 3728
- trylock
  - decaf::internal::util::concurrent::MutexImpl, 2873
- TYPE
  - inflate.h, 4627
- type
  - activemq::commands::JournalTransaction, 2311
  - activemq::commands::Message, 2613
  - activemq::commands::TransactionInfo, 3964
- TYPEDO
  - inflate.h, 4627
- uch
  - zutil.h, 4639
- uchf
  - zutil.h, 4639
- uInt
  - zconf.h, 4632
- uIntf
  - zconf.h, 4632
- ulg
  - zutil.h, 4639
- uLong
  - zconf.h, 4632
- uLongf
  - zconf.h, 4632
- uncaughtException
  - decaf::lang::Thread::UncaughtExceptionHandler, 4019
- UnknownHostException
  - decaf::net::UnknownHostException, 4020, 4021
- UnknownServiceException
  - decaf::net::UnknownServiceException, 4023, 4024
- unlock
  - activemq::core::MessageDispatchChannel, 2689
  - decaf::internal::util::concurrent::MutexImpl, 2874
  - decaf::internal::util::concurrent::SynchronizableImpl, 3824
  - decaf::io::InputStream, 2114
  - decaf::io::OutputStream, 2996
- decaf::util::AbstractCollection, 171
- decaf::util::concurrent::ConcurrentStlMap, 1280
- decaf::util::concurrent::Lock, 2451
- decaf::util::concurrent::locks::Lock, 2457
- decaf::util::concurrent::locks::ReentrantLock, 3279
- decaf::util::concurrent::Mutex, 2869
- decaf::util::concurrent::Synchronizable, 3816
- decaf::util::StlMap, 3720
- decaf::util::StlQueue, 3729
- unmarshal
  - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3094
  - activemq::wireformat::openwire::OpenWireFormat, 2983
  - activemq::wireformat::openwire::utils::BooleanStream, 865
  - activemq::wireformat::stomp::StompWireFormat, 3754
  - activemq::wireformat::WireFormat, 4091
- unmarshalList
  - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3095
- unmarshalMap
  - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3095
- unmarshalPrimitive
  - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3096
- unmarshalPrimitiveList
  - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3096
- unmarshalPrimitiveMap
  - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 3096
- unpark
  - decaf::util::concurrent::locks::LockSupport, 2461
- unread
  - decaf::io::PushbackInputStream, 3237, 3238
- unregisterFactory
  - activemq::transport::TransportRegistry, 4018
  - activemq::wireformat::WireFormatRegistry, 4132
- unsetenv
  - decaf::lang::System, 3846

- UNSUBSCRIBE
  - activemq::wireformat::stomp::StompCommandConstants, 3740
  - ush
- unsubscribe
  - activemq::cmsutil::PooledSession, 3055
  - activemq::core::ActiveMQSession, 531
  - cms::Session, 3474
  - zutil.h, 4639
- UnsupportedEncodingException
  - decaf::io::UnsupportedEncodingException, 4026, 4027
  - UTFDataFormatException
  - decaf::io::UTFDataFormatException, 4079, 4080
- UnsupportedOperationException
  - cms::UnsupportedOperationException, 4031
  - decaf::lang::exceptions::UnsupportedOperationException, 4028, 4029
  - UUID
  - decaf::util::UUID, 4083
- update
  - decaf::util::zip::Adler32, 731, 732
  - decaf::util::zip::Checksum, 1175, 1176
  - decaf::util::zip::CRC32, 1569, 1570
  - code, 1216
  - valid
  - decaf::io::FileDescriptor, 1948
- URI
  - decaf::net::URI, 4034, 4035
- URLEncoderDecoder
  - decaf::internal::net::URLEncoderDecoder, 4045
  - validate
  - decaf::internal::net::URLEncoderDecoder, 4046
- URIHelper
  - decaf::internal::net::URIHelper, 4048, 4049
  - validateAuthority
  - decaf::internal::net::URIHelper, 4051
  - validateFragment
  - decaf::internal::net::URIHelper, 4052
  - validatePath
  - decaf::internal::net::URIHelper, 4052
  - validateQuery
  - decaf::internal::net::URIHelper, 4052
  - validateScheme
  - decaf::internal::net::URIHelper, 4053
- URIParam
  - activemq::core::ActiveMQConstants, 299
  - validateSimple
  - decaf::internal::net::URLEncoderDecoder, 4046
- uriParams
  - activemq::core::ActiveMQConstants::StaticInitializer, 3693
  - validateSimple
  - decaf::internal::net::URIHelper, 4053
- uriParamsMap
  - activemq::core::ActiveMQConstants::StaticInitializer, 3693
  - validateUserinfo
  - decaf::internal::net::URIHelper, 4053
- URIPool
  - activemq::transport::failover::URIPool, 4055
  - value
  - activemq::commands::BrokerId, 878
  - activemq::commands::ConnectionId, 1368
- URISyntaxException
  - decaf::net::URISyntaxException, 4061, 4062
  - activemq::commands::ConsumerId, 1477
  - activemq::commands::LocalTransactionId, 2424
- URIType
  - decaf::internal::net::URIType, 4066
  - activemq::commands::ProducerId, 3161
  - activemq::commands::SessionId, 3480
- URL
  - decaf::net::URL, 4073
- userID
  - activemq::commands::Message, 2613
  - decaf::lang::Boolean, 860
  - decaf::lang::Byte, 977, 978
  - decaf::lang::Character, 1134
  - decaf::lang::Double, 1852
- userName

- decaf::lang::Float, 1972, 1973
- decaf::lang::Integer, 2158, 2159
- decaf::lang::Long, 2509, 2510
- decaf::lang::Short, 3547
- decaf::util::concurrent::TimeUnit, 3928
- values
  - decaf::util::concurrent::ConcurrentStlMap, 1280
  - decaf::util::concurrent::TimeUnit, 3930
  - decaf::util::Map, 2550
  - decaf::util::StlMap, 3720
- variant
  - decaf::util::UUID, 4087
- verify
  - decaf::security::cert::Certificate, 1115
- version
  - decaf::util::UUID, 4087
- visit
  - activemq::commands::BrokerError, 872
  - activemq::commands::BrokerInfo, 909
  - activemq::commands::Command, 1232
  - activemq::commands::ConnectionControl, 1306
  - activemq::commands::ConnectionError, 1335
  - activemq::commands::ConnectionInfo, 1398
  - activemq::commands::ConsumerControl, 1446
  - activemq::commands::ConsumerInfo, 1508
  - activemq::commands::ControlCommand, 1538
  - activemq::commands::DestinationInfo, 1783
  - activemq::commands::FlushCommand, 2001
  - activemq::commands::KeepAliveInfo, 2337
  - activemq::commands::Message, 2612
  - activemq::commands::MessageAck, 2647
  - activemq::commands::MessageDispatchChannel, 2682
  - activemq::commands::MessageDispatchChannelInfo, 2720
  - activemq::commands::MessagePull, 2827
  - activemq::commands::ProducerAck, 3187
  - activemq::commands::ProducerInfo, 3189
  - activemq::commands::RemoveInfo, 3287
  - activemq::commands::RemoveSubscriptionInfo, 3317
  - activemq::commands::ReplayCommand, 3346
  - activemq::commands::Response, 3382
  - activemq::commands::SessionInfo, 3508
  - activemq::commands::ShutdownInfo, 3575
  - activemq::commands::TransactionInfo, 3963
  - activemq::commands::WireFormatInfo, 4104
- voidp
  - zconf.h, 4632
- voidpc
  - zconf.h, 4632
- voidpf
  - zconf.h, 4632
- w\_bits
  - internal\_state, 2191
- w\_mask
  - internal\_state, 2191
- w\_size
  - internal\_state, 2191
- wait
  - activemq::core::MessageDispatchChannel, 2689, 2690
  - decaf::internal::util::concurrent::ConditionImpl, 1293
  - decaf::internal::util::concurrent::SynchronizableImpl, 3824, 3825
  - decaf::io::InputStream, 2115, 2116
  - decaf::io::OutputStream, 2996, 2997
  - decaf::util::AbstractCollection, 171, 172
  - decaf::util::concurrent::ConcurrentStlMap, 1280, 1281
  - decaf::util::concurrent::Mutex, 2870
  - decaf::util::concurrent::Synchronizable, 3817, 3819, 3820
  - decaf::util::StlMap, 3720, 3721
  - decaf::util::StlQueue, 3729, 3730
  - WAIT\_INFINITE
    - Concurrent.h, 4713
  - waitForSpace
    - activemq::util::MemoryUsage, 2595
    - activemq::util::Usage, 4077
  - WAITING
    - decaf::lang::Thread, 3880
  - wakeup
    - activemq::util::MemoryUsage, 2595
    - activemq::util::Usage, 4077

- activemq::core::ActiveMQSession, 531
- activemq::core::ActiveMQSessionExecutor, 4118
  - 535
- activemq::threads::CompositeTaskRunner, 4106
  - 1258
- activemq::threads::DedicatedTaskRunner, 4110
  - 1726
- activemq::threads::TaskRunner, 3850
  - WireFormatNegotiator
- activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller, 4129
- activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller, 4129
- activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller, 4129
- activemq::wireformat::WireFormatNegotiator, 4129
- want
  - gz\_state, 2041
- Warn
  - decaf::util::logging, 156
- warn
  - decaf::util::logging::SimpleLogger, 3607
- WARNING
  - decaf::util::logging::Level, 2408
- warning
  - decaf::util::logging::Logger, 2474
- was
  - inflate\_state, 2087
- wasPrepared
  - activemq::commands::JournalTransaction, 2311
- wbits
  - inflate\_state, 2087
- what
  - cms::CMSException, 1194
  - decaf::lang::Exception, 1893
- whave
  - inflate\_state, 2087
- WIN\_INIT
  - deflate.h, 4623
- window
  - inflate\_state, 2087
  - internal\_state, 2191
- window\_size
  - internal\_state, 2191
- windowSize
  - activemq::commands::ProducerInfo, 3190
- WireFormatInfo
  - activemq::commands::WireFormatInfo, 4096
- WireFormatInfoMarshaller
  - activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller, 4122
  - activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller, 4114
  - activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller, 4126
- activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller, 4118
- activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller, 4106
- activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller, 4110
- activemq::wireformat::WireFormatNegotiator, 4129
- wnext
  - inflate\_state, 2087
- work
  - inflate\_state, 2087
- wrap
  - decaf::nio::ByteBuffer, 1078
  - decaf::nio::CharBuffer, 1166
  - decaf::nio::DoubleBuffer, 1877
  - decaf::nio::FloatBuffer, 1997
  - decaf::nio::IntBuffer, 2142
  - decaf::nio::LongBuffer, 2534
  - decaf::nio::ShortBuffer, 3571, 3572
  - inflate\_state, 2087
  - internal\_state, 2191
- write
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2954
  - decaf::internal::net::tcp::TcpSocket, 3859
  - decaf::internal::util::ByteArrayAdapter, 1003
  - decaf::io::OutputStream, 2998, 2999
  - decaf::io::Writer, 4137–4139
- WRITE\_FAILURE
  - decaf::util::logging::ErrorManager, 1886
- writeBoolean
  - activemq::commands::ActiveMQBytesMessage, 229
  - activemq::commands::ActiveMQStreamMessage, 547
  - activemq::wireformat::openwire::utils::BooleanStream, 865
  - cms::BytesMessage, 1092
  - cms::StreamMessage, 3770
  - decaf::io::DataOutput, 1624
  - decaf::io::DataOutputStream, 1630
  - decaf::nio::ByteBuffer, 1078
  - decaf::nio::CharBuffer, 1166
  - decaf::nio::DoubleBuffer, 1877
  - decaf::nio::FloatBuffer, 1997
  - decaf::nio::IntBuffer, 2142
  - decaf::nio::LongBuffer, 2534
  - decaf::nio::ShortBuffer, 3571, 3572
  - inflate\_state, 2087
  - internal\_state, 2191

cms::StreamMessage, 3771  
 decaf::io::DataOutput, 1624  
 decaf::io::DataOutputStream, 1631  
 writeBytes  
   activemq::commands::ActiveMQBytesMessage, 230  
   activemq::commands::ActiveMQStreamMessage, 548, 549  
   cms::BytesMessage, 1093  
   cms::StreamMessage, 3771  
   decaf::io::DataOutput, 1624  
   decaf::io::DataOutputStream, 1631  
 writeChar  
   activemq::commands::ActiveMQBytesMessage, 231  
   activemq::commands::ActiveMQStreamMessage, 549  
   cms::BytesMessage, 1093  
   cms::StreamMessage, 3772  
   decaf::io::DataOutput, 1625  
   decaf::io::DataOutputStream, 1631  
 writeChars  
   decaf::io::DataOutput, 1625  
   decaf::io::DataOutputStream, 1631  
 WriteChecker  
   activemq::transport::inactivity::InactivityMonitor, 2069  
   activemq::transport::inactivity::WriteChecker, 4133  
 writeDouble  
   activemq::commands::ActiveMQBytesMessage, 231  
   activemq::commands::ActiveMQStreamMessage, 550  
   cms::BytesMessage, 1094  
   cms::StreamMessage, 3772  
   decaf::io::DataOutput, 1625  
   decaf::io::DataOutputStream, 1631  
 writeFloat  
   activemq::commands::ActiveMQBytesMessage, 231  
   activemq::commands::ActiveMQStreamMessage, 550  
   cms::BytesMessage, 1094  
   cms::StreamMessage, 3773  
   decaf::io::DataOutput, 1626  
   decaf::io::DataOutputStream, 1631  
 writeInt  
   activemq::commands::ActiveMQBytesMessage, 232  
   activemq::commands::ActiveMQStreamMessage, 550  
   cms::BytesMessage, 1095  
   cms::StreamMessage, 3773  
   decaf::io::DataOutput, 1626  
   decaf::io::DataOutputStream, 1631  
   decaf::util::concurrent::locks::ReadWriteLock, 3265  
 writeLong  
   activemq::commands::ActiveMQBytesMessage, 232  
   activemq::commands::ActiveMQStreamMessage, 551  
   cms::BytesMessage, 1095  
   cms::StreamMessage, 3773  
   decaf::io::DataOutput, 1626  
   decaf::io::DataOutputStream, 1631  
 Writer  
   decaf::io::Writer, 4135  
 writeShort  
   activemq::commands::ActiveMQBytesMessage, 233  
   activemq::commands::ActiveMQStreamMessage, 551  
   cms::BytesMessage, 1095  
   cms::StreamMessage, 3774  
   decaf::io::DataOutput, 1627  
   decaf::io::DataOutputStream, 1631  
 writeString  
   activemq::commands::ActiveMQBytesMessage, 233  
   activemq::commands::ActiveMQStreamMessage, 552  
   activemq::util::MarshallingSupport, 2574  
   cms::BytesMessage, 1096  
   cms::StreamMessage, 3774  
 writeString16  
   activemq::util::MarshallingSupport, 2574  
 writeString32  
   activemq::util::MarshallingSupport, 2575  
 writeTag  
   decaf::io::ByteArrayOutputStream, 1049  
 writeUnsignedShort  
   activemq::commands::ActiveMQBytesMessage, 233  
   activemq::commands::ActiveMQStreamMessage, 552  
   cms::BytesMessage, 1096  
   cms::StreamMessage, 3775



---

decaf::io::DataOutput, 1627	zlib.h, 4636
decaf::io::DataOutputStream, 1631	Z_DEFAULT_COMPRESSION
writeUTF	zlib.h, 4636
activemq::commands::ActiveMQBytesMessage, 234	Z_DEFAULT_STRATEGY
activemq::commands::ActiveMQBytesMessage, 234	zlib.h, 4636
cms::BytesMessage, 1097	Z_DEFLATED
decaf::io::DataOutput, 1627	zlib.h, 4636
decaf::io::DataOutputStream, 1631	z_errmsg
written	zutil.h, 4639
decaf::io::DataOutputStream, 1631	Z_ERRNO
wsiz	zlib.h, 4636
inflate_state, 2087	Z_FILTERED
	zlib.h, 4636
XATransactionId	Z_FINISH
activemq::commands::XATransactionId, 4144	zlib.h, 4636
	Z_FIXED
XATransactionIdMarshaller	zlib.h, 4636
activemq::wireformat::openwire::marshaller::XATransactionIdMarshaller, 4161	Z_FULL_FLUSH
	zlib.h, 4636
activemq::wireformat::openwire::marshaller::XATransactionIdMarshaller, 4152	Z_HUFFMAN_ONLY
	zlib.h, 4636
activemq::wireformat::openwire::marshaller::XATransactionIdMarshaller, 4165	Z_INTERNAL
	zlib.h, 4636
activemq::wireformat::openwire::marshaller::XATransactionIdMarshaller, 4157	Z_NO_COMPRESSION
	zlib.h, 4636
activemq::wireformat::openwire::marshaller::XATransactionIdMarshaller, 4169	Z_NO_FLUSH
	zlib.h, 4636
activemq::wireformat::openwire::marshaller::XATransactionIdMarshaller, 4148	Z_NULL
xflags	
gz_header_s, 2039	zlib.h, 4636
XMLFormatter	z_off64_t
decaf::util::logging::XMLFormatter, 4173	zconf.h, 4632
	z_off_t
yield	zconf.h, 4632
decaf::lang::Thread, 3886	Z_OK
	zlib.h, 4636
Z_ASCII	Z_PARTIAL_FLUSH
zlib.h, 4636	zlib.h, 4636
Z_BEST_COMPRESSION	Z_RLE
zlib.h, 4636	zlib.h, 4636
Z_BEST_SPEED	z_stream
zlib.h, 4636	zlib.h, 4636
Z_BINARY	Z_STREAM_END
zlib.h, 4636	zlib.h, 4636
Z_BLOCK	Z_STREAM_ERROR
zlib.h, 4636	zlib.h, 4636
Z_BUF_ERROR	z_stream_s, 4174
zlib.h, 4636	adler, 4175
Z_DATA_ERROR	avail_in, 4175

---

- avail\_out, 4175
- data\_type, 4175
- msg, 4175
- next\_in, 4175
- next\_out, 4175
- opaque, 4175
- reserved, 4175
- state, 4175
- total\_in, 4175
- total\_out, 4175
- zalloc, 4175
- zfree, 4175
- z\_stream
  - zlib.h, 4636
- Z\_SYNC\_FLUSH
  - zlib.h, 4636
- Z\_TEXT
  - zlib.h, 4636
- Z\_TREES
  - zlib.h, 4636
- Z\_UNKNOWN
  - zlib.h, 4636
- Z\_VERSION\_ERROR
  - zlib.h, 4636
- ZALLOC
  - zutil.h, 4639
- zalloc
  - z\_stream\_s, 4175
- zconf.h
  - Byte, 4632
  - Bytef, 4632
  - charf, 4632
  - const, 4632
  - FAR, 4632
  - intf, 4632
  - MAX\_MEM\_LEVEL, 4632
  - MAX\_WBITS, 4632
  - OF, 4632
  - SEEK\_CUR, 4632
  - SEEK\_END, 4632
  - SEEK\_SET, 4632
  - uInt, 4632
  - uIntf, 4632
  - uLong, 4632
  - uLongf, 4632
  - voidp, 4632
  - voidpc, 4632
  - voidpf, 4632
  - z\_off64\_t, 4632
  - z\_off\_t, 4632
  - ZEXPORT, 4632
  - ZEXPORTVA, 4632
  - ZEXTERN, 4632
  - ZEXPORT
    - zconf.h, 4632
  - ZEXPORTVA
    - zconf.h, 4632
  - ZEXTERN
    - zconf.h, 4632
  - ZFREE
    - zutil.h, 4639
  - zfree
    - z\_stream\_s, 4175
  - ZipException
    - decaf::util::zip::ZipException, 4176, 4177
  - zlib.h
    - deflateInit, 4635
    - deflateInit2, 4635
    - gz\_header, 4636
    - gz\_headerp, 4636
    - gzFile, 4636
    - inflateBackInit, 4635
    - inflateInit, 4635
    - inflateInit2, 4636
    - OF, 4636
    - Z\_ASCII, 4636
    - Z\_BEST\_COMPRESSION, 4636
    - Z\_BEST\_SPEED, 4636
    - Z\_BINARY, 4636
    - Z\_BLOCK, 4636
    - Z\_BUF\_ERROR, 4636
    - Z\_DATA\_ERROR, 4636
    - Z\_DEFAULT\_COMPRESSION, 4636
    - Z\_DEFAULT\_STRATEGY, 4636
    - Z\_DEFLATED, 4636
    - Z\_ERRNO, 4636
    - Z\_FILTERED, 4636
    - Z\_FINISH, 4636
    - Z\_FIXED, 4636
    - Z\_FULL\_FLUSH, 4636
    - Z\_HUFFMAN\_ONLY, 4636
    - Z\_MEM\_ERROR, 4636
    - Z\_NEED\_DICT, 4636
    - Z\_NO\_COMPRESSION, 4636
    - Z\_NO\_FLUSH, 4636
    - Z\_NULL, 4636
    - Z\_OK, 4636
    - Z\_PARTIAL\_FLUSH, 4636
    - Z\_RLE, 4636

- z\_stream, 4636
- Z\_STREAM\_END, 4636
- Z\_STREAM\_ERROR, 4636
- z\_streamp, 4636
- Z\_SYNC\_FLUSH, 4636
- Z\_TEXT, 4636
- Z\_TREES, 4636
- Z\_UNKNOWN, 4636
- Z\_VERSION\_ERROR, 4636
- ZLIB\_VER\_MAJOR, 4636
- ZLIB\_VER\_MINOR, 4636
- ZLIB\_VER\_REVISION, 4636
- ZLIB\_VER\_SUBREVISION, 4636
- ZLIB\_VERNUM, 4636
- ZLIB\_VERSION, 4636
- zlib\_version, 4636
- ZLIB\_INTERNAL
  - gzguts.h, 4625
  - zutil.h, 4639
- ZLIB\_VER\_MAJOR
  - zlib.h, 4636
- ZLIB\_VER\_MINOR
  - zlib.h, 4636
- ZLIB\_VER\_REVISION
  - zlib.h, 4636
- ZLIB\_VER\_SUBREVISION
  - zlib.h, 4636
- ZLIB\_VERNUM
  - zlib.h, 4636
- ZLIB\_VERSION
  - zlib.h, 4636
- zlib\_version
  - zlib.h, 4636
- zstrerror
  - gzguts.h, 4625
- zutil.h
  - Assert, 4639
  - DEF\_MEM\_LEVEL, 4639
  - DEF\_WBITS, 4639
  - DYN\_TREES, 4639
  - ERR\_MSG, 4639
  - ERR\_RETURN, 4639
  - F\_OPEN, 4639
  - local, 4639
  - MAX\_MATCH, 4639
  - MIN\_MATCH, 4639
  - OF, 4639
  - OS\_CODE, 4639
  - PRESET\_DICT, 4639
  - STATIC\_TREES, 4639
  - STORED\_BLOCK, 4639
  - Trace, 4639
  - Tracec, 4639
  - Tracecv, 4639
  - Tracev, 4639
  - Tracevv, 4639
  - TRY\_FREE, 4639
  - uch, 4639
  - uchf, 4639
  - ulg, 4639
  - ush, 4639
  - ushf, 4639
  - z\_errmsg, 4639
  - ZALLOC, 4639
  - ZFREE, 4639
  - ZLIB\_INTERNAL, 4639